DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS E INGENIERÍA DEL SOFTWARE

Facultad de Informática

Universidad Politécnica de Madrid

TESIS DOCTORAL

UNIFICACIÓN DE LOS PROTOCOLOS DE MULTIPUNTO FIABLE OPTIMIZANDO LA ESCALABILIDAD Y EL RETARDO

Autora

María Carmen Calderón Pastor

Licenciada en Informática

Director

Arturo Azcorra Saloña

Dr. Ingeniero de Telecomunicación

Año 1996

TESIS DOCTORAL

UNIFICACIÓN DE LOS PROTOCOLOS DE MULTIPUNTO FIABLE OPTIMIZANDO LA ESCALABILIDAD Y EL RETARDO

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día 25 de Septiembre de 1996.

Presidente	D. José Luis Maté Hernández
Vocal	D. Joan Vinyes Sanz
Vocal	D. Arturo Ribagorda Garnacho
Vocal	D. Juan Manuel Vozmediano Torres
Secretario	D. Javier Yagüez García

Realizado el acto de defensa y lectura de la Tesis el día 21 de Octubre de 1996.

en la Facultad de Informática de la UPM

Calificación: "Aprobado cum laude" por unanimidad

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

A mi familia

Resumen

Las aplicaciones distribuidas que precisan de un servicio *multipunto fiable* son muy numerosas, y entre otras es posible citar las siguientes: bases de datos distribuidas, sistemas operativos distribuidos, sistemas de simulación interactiva distribuida y aplicaciones de distribución de software, publicaciones o noticias. Aunque en sus orígenes el dominio de aplicación de tales sistemas distribuidos estaba reducido a una única subred (por ejemplo una Red de Área Local) posteriormente ha surgido la necesidad de ampliar su aplicabilidad a interredes.

La aproximación tradicional al problema del multipunto fiable en interredes se ha basado principalmente en los dos siguientes puntos: (1) proporcionar en un mismo protocolo muchas garantías de servicio (por ejemplo fiabilidad, atomicidad y ordenación) y a su vez algunas de éstas en distintos grados, sin tener en cuenta que muchas aplicaciones multipunto que precisan fiabilidad no necesitan otras garantías; y (2) extender al entorno multipunto las soluciones ya adoptadas en el entorno punto a punto sin considerar las características diferenciadoras; y de aquí, que se haya tratado de resolver el problema de la fiabilidad multipunto con protocolos extremo a extremo (protocolos de transporte) y utilizando esquemas de recuperación de errores, centralizados (las retransmisiones se hacen desde un único punto, normalmente la fuente) y globales (los paquetes solicitados se vuelven a enviar al grupo completo).

En general, estos planteamientos han dado como resultado protocolos que son ineficientes en tiempo de ejecución, tienen problemas de escalabilidad, no hacen un uso óptimo de los recursos de red y no son adecuados para aplicaciones sensibles al retardo.

En esta Tesis se investiga el problema de la fiabilidad multipunto en interredes operando en modo datagrama y se presenta una forma novedosa de enfocar el problema: es más óptimo *resolver el problema de la fiabilidad multipunto a nivel de red y separar la fiabilidad de otras garantías de servicio,* que pueden ser proporcionadas por un protocolo de nivel superior o por la propia aplicación.

Siguiendo este nuevo enfoque se ha diseñado un protocolo multipunto fiable que opera a nivel de red (denominado RMNP). Las características más representativas del RMNP

son las siguientes; (1) sigue una aproximación orientada al emisor, lo cual permite lograr un grado muy alto de fiabilidad; (2) plantea un esquema de recuperación de errores distribuido (las retransmisiones se hacen desde ciertos encaminadores intermedios que siempre estarán más cercanos a los miembros que la propia fuente) y de ámbito restringido (el alcance de las retransmisiones está restringido a un cierto número de miembros). Este esquema hace posible optimizar el retardo medio de distribución y disminuir la sobrecarga introducida por las retransmisiones; (3) incorpora en ciertos encaminadores funciones de agregación y filtrado de paquetes de control, que evitan problemas de implosión y reducen el tráfico que fluye hacia la fuente.

Con el fin de evaluar el comportamiento del protocolo diseñado, se han realizado pruebas de simulación obteniéndose como principales conclusiones que, el RMNP escala correctamente con el tamaño del grupo, hace un uso óptimo de los recursos de red y es adecuado para aplicaciones sensibles al retardo.

Abstract

There are many distributed applications that require a *reliable multicast* service, including: distributed databases, distributed operating systems, distributed interactive simulation systems and distribution applications of software, publications or news. Although the application domain of distributed systems of this type was originally confined to a single subnetwork (for example, a Local Area Network), it later became necessary extend their applicability to internetworks.

The traditional approach to the reliable multicast problem in internetworks is based mainly on the following two points: (1) provide a lot of service guarantees in one and the same protocol (for example, reliability, atomicity and ordering) and different levels of guarantee in some cases, without taking into account that many multicast applications that require reliability do not need other guarantees, and (2) extend solutions adopted in the unicast environment to the multicast environment without taking into account their distinctive characteristics. So, the attempted solutions to the multicast reliability problem were end-to-end protocols (transport protocols) and centralized error recovery schemata (retransmissions made from a single point, normally the source) and global error retrieval schemata (the requested packets are retransmitted to the whole group).

Generally, these approaches have resulted in protocols that are inefficient in execution time, have scaling problems, do not make optimum use of network resources and are not suitable for delay-sensitive applications.

Here, the multicast reliability problem is investigated in internetworks operating in datagram mode and a new way of approaching the problem is presented: it is better to *solve to the multicast reliability problem at network level and separate reliability from other service guarantees* that can be supplied by a higher protocol or the application itself.

A reliable multicast protocol that operates at network level (called RMNP) has been designed on the basis of this new approach. The most representative characteristics of the RMNP are as follows: (1) it takes a transmitter-oriented approach, which provides for a very high reliability level; (2) it provides for an error retrieval schema that is distributed (the retransmissions are made from given intermediate routers that will always be closer to

the members than the source itself) and of restricted scope (the scope of the retransmissions is confined to a given number of members), and this schema makes it possible to optimize the mean distribution delay and reduce the overload caused by retransmissions; (3) some routers include control packet aggregation and filtering functions that prevent implosion problems and reduce the traffic flowing towards the source.

Simulation test have been performed in order to evaluate the behaviour of the protocol designed. The main conclusions are that the RMNP scales correctly with group size, makes optimum use of network resources and is suitable for delay-sensitive applications.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1.

INTRODUCCIÓN Y OBJETIVOS

Aunque tradicionalmente el modelo de comunicación en las redes de datos ha sido el punto a punto[1], en el cual un emisor envía datos a un receptor; en los últimos años ha surgido la necesidad de que las redes también soporten de forma eficiente un modelo de *comunicación en grupo o comunicación multipunto*[2], en el cual los datos en lugar de ir dirigidos a un único receptor, son enviados a un grupo. Un grupo está formado por *m* sistemas receptores denominados *miembros* y frecuentemente es identificado de forma unívoca por una *dirección multipunto*. Los sistemas emisores que envían datos a un grupo son denominados *fuentes*. Según esto, cada vez que una fuente envía datos a un grupo, éstos son recibidos por cada uno de sus miembros.

Son muy numerosas las aplicaciones que pueden verse beneficiadas de un servicio de comunicación multipunto (denominadas aplicaciones multipunto), siendo éstas de naturaleza muy diversa. Como ejemplo pueden mencionarse las siguientes: sistemas de bases de datos distribuidas [Cha84], sistemas operativos distribuidos [Tan95, Gie92, KTHB89], sistemas de conferencia multimedia [AE92, HW95, Tur94], sistemas de simulación interactiva distribuida[3] [SS95], aplicaciones de enseñanza asistida por ordenador [PFB95], aplicaciones de trabajo cooperativo soportado por ordenador[4] [NR95], y aplicaciones de distribución de noticias, software [JSW91] o publicaciones [Don95].

---

[1] Conocido por el término anglosajón *unicast*.

[2] Conocido por el término anglosajón *multicast*.

[3] Más conocidas por *DIS (Distributed Interactive Simulation)*.

[4] Más conocidas por *CSCW (Computer Supported Cooperative Work)* o por el término anglosajón *Groupware*.

Dado el interés creciente por algunas de las aplicaciones multipunto, como por ejemplo los sistemas de conferencia multimedia o los sistemas de simulación interactiva distribuida, actualmente, la comunicación multipunto es un área de investigación que ha suscitado un gran interés, siendo numerosos los grupos de trabajo que están dedicando esfuerzos a resolver los distintos aspectos asociados a la problemática del multipunto, entre otros, la reserva de recursos, las estrategias de encaminamiento, la gestión de grupos o la calidad del servicio.

En un principio el ámbito de operación de las aplicaciones multipunto estaba normalmente restringido a entornos de área local, una o varias RALs (Redes de Área Local) interconectadas, y a grupos con un número reducido de miembros. Consecuentemente, las soluciones tradicionales al problema de la comunicación multipunto están construidas sobre estos supuestos y tienen una escalabilidad limitada.

Más recientemente ha surgido la necesidad de implantar estas aplicaciones multipunto en redes de área extensa, y en general en interredes, y de posibilitar que el número de miembros sea elevado. En el contexto de interredes de área extensa los recursos no son tan abundantes como en una RAL, y consecuentemente toda solución aplicable a este entorno debe tener entre uno de sus principales objetivos la optimización del uso de recursos de red, condicionante normalmente obviado por las aproximaciones tradicionales.

Últimamente muchos investigadores se han centrado en las interredes. Fruto de este interés y como resultado de diversos trabajos, han sido propuestos distintos protocolos que proporcionan un servicio multipunto en interredes basadas en datagramas (el IP multipunto [Dee89] y el CLNP[5] multipunto [Mar95]), y desde principios de 1992 se ha puesto en funcionamiento de forma experimental, una interred que opera en modo datagrama y ofrece un servicio multipunto, ésta es denominada la MBone[6] [Eri94, MB94, TCD95].

La MBone es una red multipunto "virtual", construida sobre la Internet, que utiliza el IP multipunto. El primer uso de la MBone fue difundir las reuniones del IETF[7] de la Internet [CD92] utilizando aplicaciones de conferencia multimedia experimentales. La MBone está creciendo rápidamente y el número de usuarios que utilizan sus servicios, para experimentar con las más diversas aplicaciones multipunto está aumentando de día en día.

[5] *ConnectionLess mode Nerwork Protocol.*

[6] *Internet Multicast Backbone.*

[7] *Internet Engineering Task Force.*

Dado el amplio rango y la naturaleza tan diversa de las aplicaciones multipunto, también son muy dispares las garantías o calidades de servicio que éstas precisan. Entre las numerosas garantías de servicio aplicables en el entorno multipunto, la *fiabilidad* ocupa un lugar muy importante. Esta relevancia se debe a que la gran mayoría de las aplicaciones multipunto precisan un servicio fiable en mayor o menor grado, y las dificultades técnicas que conlleva lograr dicho servicio:

- ❒ En el entorno multipunto la problemática de la fiabilidad adquiere nuevos retos y dimensiones inexistentes en el entorno punto a punto, entre otros los siguientes: ¿cómo ofrecer un servicio fiable que escale adecuadamente con relación al número de miembros del grupo?, ¿cómo evitar que se produzca un incremento excesivo del retardo de distribución al aumentar, el número de miembros o la distancia geográfica entre los distintos participantes de la comunicación?.

- ❒ En el entorno multipunto surgen nuevas semánticas de la fiabilidad. Con el fin de clarificar esta afirmación se muestran seguidamene dos casos:

  - ♦ Un grupo está formado por n miembros, y de aquí que además de la semántica "los datos han sido recibidos por *todos* los miembros", sea posible definir ciertos grados de fiabilidad menos exigentes; por ejemplo, una fiabilidad grado k (siendo $k < n$) que garantiza que de los n miembros del grupo, k han recibido todos los datos.

  - ♦ La actuación lógica ante el fallo de un miembro o una partición en la interred, depende de la semántica de la aplicación. Por ejemplo, dada una aplicación de distribución de software que tiene como destino 500 puntos distintos, ¿es lógico que ante una partición en la interred que causa que dos puntos receptores queden aislados, sea abortada la comunicación?.

El problema de la fiabilidad multipunto en interredes constituye una línea de investigación aún abierta y de un gran interés. El hecho de ser una línea aún abierta es debido a que las aproximaciones actuales plantean las siguientes deficiencias:

- ❒ *No son adecuadas cuando el número de miembros es muy elevado.* Uno de los problemas típicos surge cuando n miembros intentan contestar, casi simultáneamente a la fuente (por ejemplo para confirmar que recibieron correctamente los datos enviados), produciendo una saturación de dicha fuente y una concentración de tráfico en las líneas que conducen a ésta. Por otro lado, al aumentar el número de miembros se incrementa la probabilidad de que al menos un miembro pierda cada

uno de los paquetes distribuidos por la fuente, hecho que causa un aumento del retardo medio de distribución. El retardo medio de distribución puede definirse como, el tiempo transcurrido desde que un paquete es liberado por la fuente hasta que es recibido correctamente por *todos* los miembros del grupo.

- *No son adecuadas cuando la interred es muy extensa*, con esta afirmación se quiere significar el hecho de que los componentes de la comunicación (fuentes y miembros) estén muy alejados geográficamente. Uno de los problemas típicos que surgen al incrementarse el tamaño de la interred, es que debido al aumento en el retardo de tránsito por la interred y en la probabilidad de pérdida de paquetes, el retardo medio de distribución sufre un incremento considerablemente, siendo inaceptable para ciertas aplicaciones.

- *No son genéricas*. Con esta afirmación se quiere significar el hecho de que muchas de las aproximaciones actuales intentan satisfacer los requisitos de una aplicación multipunto particular. Sin embargo, existe tal número de posibles aplicaciones multipunto y con necesidades tan diversas, que no parece adecuado ir proponiendo un protocolo que resuelva las necesidades concretas de cada aplicación, sino que es más óptimo plantear una solución genérica que pueda ser utilizada por un número muy alto de aplicaciones, y que sean éstas las que mejoren el servicio ofrecido, para adecuarlo a sus propias necesidades, incorporando los mecanismos precisos.

- *No optimizan el uso de recursos de red.* Este aspecto es muy relevante en el entorno multipunto donde el número de miembros puede ser muy elevado y consiguientemente cada comunicación puede precisar una gran cantidad de recursos. Adicionalmente la optimización de recursos de red es primordial en interredes extensas, que están caracterizadas por la existencia de enlaces de largo alcance que tienen asociado un coste de transmisión elevado.

1.1 Objetivos previstos

El objetivo de esta Tesis Doctoral es contribuir a la resolución del problema de la fiabilidad multipunto en el entorno de interredes basadas en datagramas. Con el mencionado fin se pretenden analizar y considerar las peculiaridades que imponen los hechos de que un grupo esté formado por un número muy elevado de miembros y que la localización geográfica de éstos sea muy dispersa, así como las consecuencias que esto conlleva para lo-

grar una solución que haga un uso óptimo de los recursos de red y que sea aplicable a un amplio rango de aplicaciones y entre éstas las sensibles al retardo.

Para lograr el objetivo anterior se han planteado los siguientes objetivos parciales:

- **Estudio de diversas facetas de la comunicación multipunto.** En concreto se prestará una atención especial a los siguientes puntos:
  - Garantías de servicio solicitadas más frecuentemente por las aplicaciones multipunto.
  - Semánticas de grupo y modelo de servicio multipunto con más amplia aceptación.
  - Principales propuestas al problema del encaminamiento multipunto para interredes basadas en datagramas.
  - Semántica de la fiabilidad en el entorno multipunto.
- **Estudio y análisis de las aproximaciones actuales** al problema de la fiabilidad multipunto identificando las posibles deficiencias.
- **Diseño de un protocolo que ofrezca un servicio multipunto fiable.** En esta fase de diseño se irán analizando distintas alternativas para resolver cada uno de los aspectos del protocolo.
- **Análisis comparativo** del protocolo diseñado con respecto a otros que tratan de resolver la misma problemática.
- **Simulación del protocolo** diseñado y análisis de los resultados obtenidos.

La principal contribución original de esta Tesis será el diseño de un protocolo que ofrezca un servicio multipunto fiable para interredes que operan en modo datagrama y exhiba las siguientes cualidades: (1) sea genérico (aplicable a un gran número de aplicaciones y entre éstas a las sensibles al retardo); (2) tenga un comportamiento adecuado al aumentar el tamaño del grupo y el tamaño de la interred y; (3) haga un uso óptimo de los recursos de red.

1.2 Estructura de la memoria

La memoria de esta Tesis se estructura como sigue:

- En el capítulo 2, se presenta el estado del arte de los principales aspectos asociados al entorno multipunto que están relacionados con, o tienen incidencia en, el desarrollo de esta Tesis. En primer lugar, se hace una breve clasificación de las principales aplicaciones que podrían verse beneficiadas por un servicio multipunto; así mismo, se exponen las calidades de servicio precisadas normalmente por dichas aplicaciones. Posteriormente, se plantean las soluciones que han sido propuestas y están siendo investigadas por los grupos más significativos en este área de investigación. Por último, se revisa el estado del multipunto a nivel de red, presentándose el modelo de servicio planteado por Deering, así como las principales estrategias de encaminamiento multipunto para interredes basadas en datagramas.

- En el capítulo 3, se analiza cual ha sido la aproximación tradicional para resolver el problema de la fiabilidad multipunto, identificándose las principales deficiencias; y tras esto, se replantea el problema de acuerdo con una serie de objetivos a perseguir.

- El capítulo 4, contiene un estudio de la semántica de la fiabilidad en el ámbito multipunto, identificándose nuevos retos inexistentes en el entorno punto a punto.

- El capítulo 5 describe el proceso de diseño del RMNP, principal contribución original de este trabajo. A lo largo del capítulo se van exponiendo diversas alternativas para resolver los distintos aspectos del protocolo, discutiendo la idoneidad de éstas para lograr los objetivos planteados.

- El capítulo 6 se centra en los experimentos de simulación realizados, presentando la herramienta empleada, la topología de red simulada, los parámetros utilizados y demás aspectos asociados con dicha simulación. Adicionalmente, se muestran e interpretan los resultados obtenidos.

- En el capítulo 7 se realiza un análisis comparativo del RMNP, con otros trabajos que tratan de resolver el mismo problema. Con este fin, se comparan los mecanismos fundamentales del protocolo.

- En el capítulo 8 se exponen las conclusiones del trabajo desarrollado, así como una serie de líneas de continuación del mismo.

- En el apéndice A se presentan las unidades de datos del protocolo, exponiéndose el formato y la utilización de cada uno de los campos.

- Por último, en el apéndice B se detalla la operación del protocolo, y en concreto de los encaminadores RMNP, mediante un pseudocódigo.

Capítulo 2.

ESTADO DEL ARTE

Existe un amplio espectro de aplicaciones que pueden verse beneficiadas por un servicio multipunto (denominadas genéricamente *"aplicaciones multipunto"*); éstas son de naturaleza muy diversa y de aquí que sus requisitos de servicio también sean muy dispares. Estas aplicaciones pueden agruparse del siguiente modo:

- ***Sistemas Distribuidos***. En este grupo se incluyen aplicaciones tales como Sistemas Operativos Distribuidos [KTHB89, Gie92, Tan95] y Bases de Datos Distribuidas [Cha84]; áreas que están siendo investigadas desde mitad de la década de los ochenta y fueron pioneras en plantear la necesidad de un servicio de comunicación en grupo. Adicionalmente, existen otras aplicaciones de más reciente aparición, tales como los Sistemas de Simulación Interactiva Distribuida [SS95], área de investigación muy activa en la actualidad, también incluidas en este grupo.

- ***Aplicaciones multiusuario.*** Este grupo puede subdividirse de la siguiente manera:

  - *Trabajo Cooperativo Soportado por Ordenador.* En la actualidad los ordenadores están empezando a jugar nuevos roles en la comunicación interpersonal, ayudando a las personas a acceder e intercambiar información mientras trabajan en equipo [SFB+87, EGR91, Pen93, NR95]. En este subgrupo de aplicaciones se incluyen herramientas compartidas (como por ejemplo, editores compartidos) y en general, sistemas que facilitan las tareas derivadas del trabajo en grupo.

  - *Sistemas de Conferencia Multimedia.* Estos sistemas permiten a los usuarios interactuar entre ellos intercambiándose información en distintos medios [AE92]. Más en concreto, normalmente soportan conversaciones en tiempo real

a través de un intercambio coordinado de voz y vídeo; y adicionalmente, permiten el uso compartido de una aplicación tipo block de notas o pizarra compartida[1], de modo que, los participantes de la conferencia pueden compartir información multimedia viéndola simultáneamente y actualizando porciones de ésta.

De todas las aplicaciones multipunto éstas son, con diferencia, las que en la actualidad están recibiendo una mayor atención, y de aquí que existan un gran número de proyectos donde se están investigando distintos aspectos de los Sistemas de Conferencia Multimedia (por ejemplo, MICE ESPRIT 7602 [HW95] y MERCI). En la actualidad, ya existen sistemas de este tipo que están siendo usados de forma experimental en la MBone [MB94], y entre éstos se tienen: aplicaciones vídeo-audio como la *ivs* [ivs, Tur94], aplicaciones de vídeo como la *nv* [nv], de audio como la *vat* [vat] y la *NEVOT* [NEVOT] y pizarras compartidas como la *wb* [wb].

- ♦ *Enseñanza a distancia y Enseñanza asistida por ordenador.* Herramientas que permiten la educación a distancia y mediante las cuales los distintos participantes pueden interactuar, establecer conversaciones, y compartir información usando una gran variedad de herramientas multimedia [PFB95].

- ❒ ***Sistemas de control.*** En este grupo se incluyen distintas aplicaciones de control remoto tales como por ejemplo sistemas de control de tráfico aéreo.

- ❒ ***Sistemas de distribución.*** En este grupo se incluyen aplicaciones como distribución de software [JSW91], de publicaciones [Don95], noticias, vídeo bajo demanda, o valores bursátiles.

- ❒ ***Aplicaciones de búsqueda de información*** [KIFU95] ***y descubrimiento de recursos.*** En este tipo de aplicaciones el conjunto de recursos que proporcionan un mismo servicio (de impresión, de información, etc.) pertenecen a un mismo grupo multipunto.

Como puede observarse, las aplicaciones multipunto mencionadas son muy distintas en naturaleza y de aquí, que sus necesidades sobre las cualidades del servicio también difieran ampliamente. En la siguiente sección se exponen los requisitos de calidad del servicio multipunto más usuales.

[1] Conocidas por el término *Whiteboards*.

2.1 Calidad del servicio multipunto

En el entorno multipunto, el tema de la calidad del servicio adquiere nuevas dimensiones, haciéndose más complejo su análisis. Por ejemplo, aparecen nuevos niveles de fiabilidad dependiendo del modelo de grupo elegido (ver Capítulo 4), tipos distintos de ordenación no existentes en la comunicación punto a punto, y adicionalmente, ciertos aspectos como el de la escalabilidad adquieren una relevancia especial.

2.1.1 Ordenación

Algunas aplicaciones pueden verse beneficiadas si el servicio multipunto ofrece ciertas garantías en cuanto al orden en que serán recibidos los mensajes en los procesos destino. Para determinadas aplicaciones distribuidas (por ejemplo, Bases de Datos Distribuidas), la recepción de determinados mensajes en orden diferente en los distintos destinos, puede conducir a inconsistencias o problemas de bloqueo; de aquí que, el hecho de que el servicio multipunto ofrezca ciertas propiedades de ordenación pueda simplificar en gran medida el diseño de software distribuido [BJ87, ISIS]. Esta propiedad de ordenación puede ser más o menos restrictiva dando lugar a diferentes tipos de ordenación:

- *Ordenación con respecto a cada fuente.* Si los mensajes m_1 y m_2 han sido originados en la misma fuente y tienen como destino el mismo grupo, entonces todos los procesos destino los recibirán en el mismo orden relativo.

- *Ordenación de los mensajes si provienen de múltiples fuentes.* Si los mensajes m_1 y m_2 van dirigidos al mismo grupo multipunto entonces todos los procesos destino los recibirán en el mismo orden relativo, aunque dichos mensajes provengan de fuentes diferentes. Este tipo de ordenación es denominado *orden total.* El orden total no garantiza que m_1 fue generado antes que m_2, esto es imposible de determinar sin disponer de relojes sincronizados globalmente.

- *Ordenación de los mensajes que van dirigidos a múltiples grupos.* Si los mensajes m_1 y m_2 son distribuidos a dos procesos destino, dichos mensajes serán entregados a ambos procesos en el mismo orden relativo, aunque provengan de fuentes diferentes y vayan dirigidos a grupos multipunto distintos, que se solapan[2].

[2] Dos grupos multipunto se solapan si tienen miembros en común.

Estas tres clases de servicio de ordenación tienen un orden creciente de complejidad y cada clase engloba a la anterior. Otro tipo de servicio de ordenación totalmente distinto es el *orden causal*. Esta ordenación se basa en la relación "ocurrió antes" definida por Lamport [Lam78]. El concepto de un suceso "ocurrió antes" que otro en un sistema distribuido permite definir un orden parcial de sucesos.

El orden causal es el cierre transitivo de las dos siguientes relaciones:

- m_1 precede a m_2, si el mismo proceso envió m_1 antes de enviar m_2.
- m_1 precede a m_2, si m_1 fue entregado al emisor de m_2, antes de que m_2 fuera enviado.

Un servicio que ofrece orden causal garantiza que, si el mensaje m_x precede causalmente a m_y, entonces m_x será entregado antes que m_y en todos los procesos que reciben m_x y m_y.

Renesse [Ren93] hace un buen análisis del problema del orden causal presentando ejemplos de aplicaciones, de las áreas de programación paralela y consistencia de datos distribuidos, cuyo diseño se simplifica si se dispone de una garantía de orden causal. Del mismo modo, en [VRB89] se plantea como aplicaciones tolerantes a fallos también pueden beneficiarse del orden causal. Muchas de estas aplicaciones precisan un orden causal y sin embargo no requieren un orden total. Ésta es una característica importante dado que un protocolo puede ofrecer orden causal sin tener que introducir mensajes extra. Naturalmente, el proporcionar un orden causal tiene un coste, y de hecho en la mayoría de las propuestas que plantean un orden causal, se incrementa substancialmente el tamaño de los mensajes al introducir campos extra en la cabecera.

Existe diversa literatura [BCG91, APR93, HW95] dónde se analiza las necesidades de ordenación en aplicaciones de distintas áreas.

2.1.2 Atomicidad

Otra propiedad útil a la hora de diseñar sistemas distribuidos es la *atomicidad ante fallos* [RL93], esto es, garantizar la distribución de un mensaje a todos los procesos o a ninguno, incluso aunque ciertos procesos o procesadores fallen durante la distribución multipunto. Los algoritmos de grupos de procesos se simplifican mucho al no tener que considerar que un suceso improbable, como el fallo de un sistema, pueda dar lugar a una distri-

bución parcial de un mensaje. Normalmente, cuando falla un miembro del grupo, se garantiza que el resto de procesos recibirán la notificación del fallo sólo después de haber recibido todos los mensajes anteriores enviados por el proceso que ha fallado, y esto se consigue respetando el orden causal.

Chang [Cha84] analiza como la atomicidad ante fallos puede simplificar el diseño de sistemas distribuidos, y en particular, muestra su utilidad para resolver los problemas de control de concurrencia y recuperación ante fallos.

Otro tipo de atomicidad llamada *atomicidad de pertenencia* ha sido definida en [BCG91]. Ésta garantiza que los cambios de pertenencia al grupo (entradas y salidas) están totalmente ordenados y sincronizados con el resto de los mensajes. Si no se proporciona atomicidad de pertenencia nunca se sabe exactamente, cuantos procesos han recibido un mensaje multipunto en particular.

2.1.3 Escalabilidad

Esta característica del servicio adquiere una especial relevancia en el entorno multipunto. La escalabilidad puede definirse de manera informal, como una medida de lo adecuado que es el servicio multipunto ante el aumento de un determinado parámetro (por ejemplo, número de miembros, número de fuentes, etc.). Dependiendo del parámetro que varía, la escalabilidad adquiere diversas dimensiones que deben ser consideradas de forma independiente:

- *Escalabilidad ante miembros.* Es una medida del número de miembros que pueden pertenecer al mismo grupo sin que se degrade el rendimiento de la comunicación multipunto.

  Un problema típico que limita la escalabilidad ante miembros, en los protocolos de transporte multipunto fiables, es el de la **implosión** [CP88]. Este problema surge cuando los *n* miembros de un grupo envían casi al mismo tiempo una contestación a la fuente (asentimientos positivos, asentimientos negativos, etc.) saturando a ésta y produciendo una gran concentración de tráfico en el área cercana a dicha fuente.

- *Escalabilidad ante fuentes.* Es una medida del número de fuentes que pueden estar mandando información al mismo grupo sin que se degrade el rendimiento de la comunicación multipunto.

Éste es un típico problema de algunos algoritmos de encaminamiento multipunto que están diseñados de forma que, la cantidad de información de estado que es necesario mantener en la interred es proporcional al número de fuentes que tiene el grupo. Por el contrario, existen otros algoritmos de encaminamiento multipunto que escalan bien con el número de fuentes, ya que mantienen en los nodos de la interred información por grupo (independientemente del número de fuentes).

- *Escalabilidad ante el tamaño de la interred.* Es una medida del tamaño que puede tener la interred sin que se degrade el rendimiento de la comunicación multipunto.

  Entre otros, un problema típico que limita esta dimensión de la escalabilidad es el surgido al proporcionar un servicio multipunto fiable, cuando al aumentar el tamaño de la interred se incrementa de forma considerable el retardo medio de distribución, llegando incluso a ser inaceptable para determinadas aplicaciones. El *retardo medio de distribución* se define, como el retardo medio desde que los paquetes son liberados por la fuente hasta que son recibidos correctamente por todos los miembros.

- *Escalabilidad ante grupos.* Es una medida del número de grupos que pueden estar simultáneamente activos en una interred, sin que se degrade el rendimiento de la comunicación multipunto. Por ejemplo, un factor que influye directamente en que los algoritmos de encaminamiento multipunto escalen bien ante el número de grupos o no, es la cantidad de información de estado que es necesario almacenar en la interred por cada grupo activo.

La propiedad de la escalabilidad en sus distintas dimensiones ha recibido considerable atención en los últimos años por los investigadores de las áreas de, encaminamiento multipunto [DEF+94, Bal95] y multipunto fiable [JSW91, BOG+94, FJM+95].

2.1.4 Fiabilidad

La mayoría de las aplicaciones expuestas al principio del capítulo precisan que el servicio multipunto ofrecido sea, en mayor o menor grado, fiable; por ejemplo, en el caso de las aplicaciones multimedia las necesidades de fiabilidad dependen del tipo de fuente, algunas requieren transmisión fiable mientras que otras pueden tolerar una pérdida de paquetes por debajo de determinados umbrales. Dada esta necesidad, el aspecto de la fiabilidad multipunto ha recibido una considerable atención y, entre otros muchos, se pueden citar los siguientes trabajos [CM84, CP88, CW89, GS91, LB91, AFM92, APR93].

El problema de la fiabilidad en el entorno multipunto plantea una serie de retos importantes que no surgen en el ámbito punto a punto: ¿Cómo proporcionar fiabilidad sin causar problemas de implosión en la fuente?, ¿Qué grado de fiabilidad es posible garantizar cuando la fuente no conoce a priori la identidad de los miembros del grupo?, ¿Cómo ofrecer un servicio fiable optimizando el uso de los recursos de la interred?, ¿Cómo proporcionar fiabilidad en una interred muy extensa sin que aumente excesivamente el retardo de distribución?

Las múltiples propuestas que tratan de resolver el problema de la fiabilidad pueden ser clasificadas según la técnica que utilizan para proporcionar dicha fiabilidad en dos categorías: **orientadas al emisor** y **orientadas al receptor.**

Cuando se sigue una aproximación **orientada al emisor** [CP88, Raj92, Bir93, Xpr95] es responsabilidad de éste comprobar que todos los receptores obtienen una copia de cada paquete enviado. En este caso, los receptores envían al emisor asentimientos positivos (ACKs), confirmando la recepción de paquetes, y el emisor utiliza temporizadores con el propósito de detectar las posibles pérdidas. Los primeros trabajos propuestos en el área de fiabilidad multipunto eran orientados al emisor. Los principales problemas de este tipo de soluciones son:

- Puede producirse un problema de implosión en el emisor, cuando los receptores intentan devolver casi simultáneamente un asentimiento positivo y consecuentemente el emisor debe recibir y procesar un gran número de paquetes en un periodo de tiempo muy corto; obviamente, esta problemática se agrava al aumentar el tamaño del grupo . Un análisis de este efecto aparece en [Dan89].

- Como resultado de los asentimientos positivos enviados casi simultáneamente por los receptores hacia el emisor, se incrementa el tráfico en encaminadores y enlaces que conducen hacia éste. Esto incluso puede causar un problema de congestión en esta parte de la red, aumentando la probabilidad de pérdida de ACKs.

- El emisor debe mantener información de estado asociada con cada receptor. Esto tiene dos consecuencias: (1) si el número de receptores es elevado puede ser muy costoso para un único sistema mantener dicha información de estado y (2) obliga a que el emisor conozca la identidad de todos los receptores.

Cuando se sigue una aproximación **orientada al receptor** es responsabilidad de cada uno de éstos detectar la pérdida de paquetes y consecuentemente informar al emisor, utilizando asentimientos negativos (NAKs), cuando precisa la retransmisión de un paquete. No

se envían asentimientos positivos. Utilizando esta técnica surgen algunos problemas, la mayoría de ellos debido a que en operación normal la fuente no recibe ninguna información sobre el estado de los receptores:

- El emisor nunca conoce con certeza que cada destino ha recibido un paquete dado. Como consecuencia, si el protocolo ha de ser considerado totalmente fiable el emisor debería mantener indefinidamente una copia de cada paquete enviado, o al menos para lograr un grado de fiabilidad aceptable deben mantenerse durante un largo periodo de tiempo.

- La pérdida de un paquete no es detectada hasta que un segundo es recibido con éxito (paquete que será recibido fuera de secuencia). La consecuencia de esto es, que los receptores pueden necesitar esperar un tiempo ilimitado para detectar la pérdida del último paquete de una ráfaga. Esto incrementa el retardo medio de distribución, porque mientras un miembro no detecta la pérdida de un paquete no puede solicitar su retransmisión.

- El emisor no detecta el fallo de los receptores. Cuando el emisor no recibe ningún asentimiento negativo asume que todo marcha correctamente, mientras que la realidad puede ser que uno o más receptores han fallado.

- Es complejo incorporar mecanismos de control de flujo.

- Si el número de receptores es muy alto, y todos detectan la pérdida de un paquete y consecuentemente envían un paquete NAK hacia el emisor, pueden provocar una implosión de NAKs. Aunque esto es posible, es menos probable que una implosión de ACKs en las aproximaciones orientadas al emisor.

La gran mayoría de las propuestas orientadas al receptor no siguen de forma pura esta filosofía, sino que para resolver los problemas planteados incorporan, junto al empleo de asentimientos negativos, otros mecanismos adicionales. En este tipo de **soluciones orientadas al receptor mixtas** las técnicas suplementarias más usuales son las siguientes:

- *Saturación* [JSW91], esta técnica tiene como finalidad incrementar la probabilidad de una transmisión fiable y consiste en enviar varias copias del mismo paquete.

- *Supresión de NAKs* [FJM+95, YGS95, CK96], esta técnica consiste en retardar la transmisión de NAKs utilizando temporizadores aleatorios, mientras se espera que otro receptor genere un NAK (solicitando los mismos datos). Esto tiene como fina-

lidad evitar una implosión de NAKs, por lo que obliga a que los NAKs y las correspondientes retransmisiones sean enviadas en modo multipunto.

- *Paquetes de estado periódicos* [ES88, FJM+95, HSC95]. Estos paquetes de estado pueden ser enviados por el emisor o por los receptores; y en general, tienen como finalidad detectar posibles fallos de sistemas (caídas del emisor o de algún receptor) y/o marcar un límite superior al tiempo que se tarda en detectar la pérdida de un paquete. Estos paquetes de estado irán asociados a temporizadores que pueden ser fijos o variables.

- *Sondeos periódicos* [RJ87]. Esta técnica obliga a que la fuente sondee periódicamente a los receptores con la finalidad de monitorizar su estado. Esto plantea el problema de no escalar bien con respecto al número de miembros.

- *División en dos subsistemas.* La idea básica de esta técnica es la siguiente: en un sistema de comunicaciones multipunto se tienen muchas fuentes y muchos destinos; el objetivo es hacer que este sistema aparezca como dos sistemas más simples, uno con un único emisor y otro con un único receptor, esto se consigue haciendo pasar todos los paquetes por un receptor primario (llamado *"token site"*). El sistema opera de forma orientada al emisor entre las fuentes y el receptor primario, y como un sistema orientado al receptor entre el receptor primario y el resto de receptores. Esta idea fue propuesta inicialmente por Chang y Maxemchuk [CM84] y ha sido adoptada posteriormente por otros protocolos [WMK94].

En este apartado se ha hecho una primera aproximación al problema de la fiabilidad, aspecto que es tratado en más detalle en el Capítulo 4, analizándose entre otras cosas, los grados de fiabilidad factibles dependiendo del modelo de grupo elegido.

2.1.5 Gestión de grupos

Este elemento del servicio multipunto agrupa funciones para crear y borrar grupos, así como para unirse dinámicamente o dejar un grupo. Adicionalmente a estas funciones principales pueden existir otras adicionales que, entre otras cosas, permitan monitorizar las propiedades de un grupo y su estado, o averiguar los grupos activos en un momento dado.

Este aspecto del servicio multipunto se encuentra todavía en un estado muy inmaduro, aunque existen algunas propuestas, como la de Braudes y Zabele [BZ93] que plantea una Autoridad de Grupos Multipunto con una estructura jerárquica, similar a la del Sistema de

Nombres de Dominio (DNS) de la Internet, que soporte la gestión del espacio de direcciones multipunto, mantenga un registro de servicios multipunto ofrecidos en la Internet y permita gestionar los grupos.

El protocolo utilizado en Internet para la gestión de grupos es el IGMP (*Internet Group Management Protocol*) [Dee89] e incorpora funciones para dinámicamente unirse o dejar un grupo.

2.1.6 Otras calidades del servicio

Algunas aplicaciones multipunto son al mismo tiempo aplicaciones en tiempo real (por ejemplo, la conferencia multimedia o el vídeo bajo demanda). Este tipo de aplicaciones necesitan ciertas garantías de QoS[3] tales como: un retardo o una variabilidad del retardo con estrechos límites, o un caudal instantáneo mínimo aceptable; y sin embargo, tradicionalmente las arquitecturas de interred basadas en datagramas no ofrecían estas garantías. De aquí que recientemente se esté realizando un considerable esfuerzo de investigación para desarrollar arquitecturas de red que soporten de forma eficiente servicios en tiempo real (denominadas *arquitecturas de red con servicios integrados* [BCS94]).

Estas garantías asociadas a las aplicaciones en tiempo real están fuera del ámbito de este trabajo, pero al lector interesado se le recomienda consultar la siguiente bibliografía [ZDE+93, FBZ94, MS94, DB95, BCDB95].

2.2 Grupos de trabajo en multipunto fiable más significativos

La literatura es rica en arquitecturas para multipunto fiable [MTP] y en la bibliografía pueden encontrarse numerosas referencias. En esta sección se exponen las propuestas realizadas por los actuales grupos de trabajo más significativos en este área. En ningún momento se intenta realizar una exposición exhaustiva de dichos trabajos, sino plantear los puntos claves de cada uno de éstos, facilitándose referencias donde encontrar información adicional.

[3] *Quality of Service.*

2.2.1 SRM

Este protocolo está siendo diseñado en el centro de investigación LBL (*Lawrence Berkeley Laboratory*) en colaboración con personal de Xerox y de la Universidad del Sur de California. Este grupo de trabajo está integrado, entre otros, por Van Jacobson (LBL) investigador que ha realizado numerosas contribuciones en el área del encaminamiento multipunto; Steven McCanne (LBL) que ha diseñado e implementado conjuntamente con Jacobson algunas de las aplicaciones multipunto [vat, wb] más utilizadas actualmente en la MBone; y Lixia Zhang (Xerox) que ha jugado un papel muy activo en el área de protocolos de reserva para aplicaciones multipunto en tiempo real. Dada la relevancia de algunos de los integrantes de este grupo de investigación, es preciso tener en consideración los trabajos que están realizando en el área del multipunto fiable.

En la actualidad este grupo está trabajando en el diseño de un protocolo multipunto fiable que funciona a nivel de transporte, el SRM (*Scalable Reliable Multicast*) [FJM+95]. Este protocolo ha sido inicialmente desarrollado para soportar la aplicación de pizarra compartida del LBL (denominada *wb* [wb]), aunque no se descarta que su aplicabilidad pueda extenderse a otras aplicaciones multipunto.

El SRM es un protocolo que, proporciona un servicio multipunto fiable sobre un servicio de distribución de datagramas no fiable (por ejemplo el IP multipunto), no ofrece ninguna garantía de ordenación, y sigue el modelo de Deering (ver sección 2.3.1).

Para proporcionar un servicio fiable, el SRM sigue una aproximación orientada al receptor, hecho que implica que cada miembro del grupo es responsable individualmente de detectar las posibles pérdidas y de solicitar las correspondientes retransmisiones. Las pérdidas se detectan al descubrir un salto en los números de secuencia, pero dada la posibilidad de que se pierda el último paquete, cada miembro envía periódicamente en modo multipunto *mensajes de sesión,* que anuncian cuales son las fuentes activas y cual ha sido el último paquete recibido desde cada una de éstas. Estos *mensajes de sesión,* además del estado de la recepción, contienen unas marcas de tiempo que sirven para estimar la distancia (en tiempo) desde cada miembro al resto de los miembros. Cuando un receptor detecta que le faltan datos, antes de enviar un asentimiento negativo, espera un tiempo aleatorio determinado por su distancia a la fuente original de dichos datos. Del mismo modo que los datos originales, los asentimientos negativos y las retransmisiones son enviados en modo multipunto al grupo completo; consecuentemente, aunque varios miembros pueden perder un mismo paquete es probable que el más cercano al punto de fallo sea el único que solicite su retransmisión. Otros miembros, que también han perdido los mismos datos, al recibir

dicha solicitud de retransmisión suprimen su propia solicitud, impidiendo una implosión de asentimientos negativos. Cualquier miembro que tenga copia de los datos solicitados puede satisfacer la petición; sin embargo, antes de comenzar la retransmisión activa un temporizador (denominado *temporizador de reparación*), que es inicializado a un valor aleatorio dependiente de su distancia al emisor del asentimiento negativo, cuando dicho temporizador expira, dicho miembro retransmite los datos solicitados en modo multipunto. Otros miembros que también habían activado el *temporizador de reparación* (tenían los datos solicitados), al recibir las retransmisiones detienen dicho temporizador (este comportamiento impide que se produzca una implosión de respuestas).

Las principales críticas a este protocolo son las siguientes: (1) al realizar todas las solicitudes de retransmisión y las correspondientes retransmisiones (algunas veces redundantes) en modo multipunto, no se optimiza el uso de recursos de red. Como se ha visto, uno o más miembros envían la solicitud de retransmisión al grupo entero y uno o más miembros, que tienen los datos solicitados, hacen la retransmisión al grupo completo, incluso aunque la pérdida que puede haber sido causada por una congestión, esté restringida a una pequeña región de la topología del grupo; y (2) con el fin de evitar una implosión de asentimientos negativos o de retransmisiones, tanto el miembro que hace la solicitud como el miembro que hace la retransmisión, retardan estos procesos en un tiempo aleatorio. Este comportamiento aumenta el tiempo de recuperación de los paquetes perdidos, incrementándose con ello el retardo medio de distribución.

2.2.2 XTP Forum

El *XTP Forum* es un foro internacional con representantes de los sectores industrial, gubernamental, militar y universitario[4]. El objetivo de este foro es promover el diseño, implementación y uso de un protocolo que satisficiera las necesidades de un amplio rango de aplicaciones y que sea capaz de operar sobre distintas tecnologías de red. El *Xpress Transport Protocol*[5] (XTP) [Xpr95] satisface los requisitos de una gran variedad de aplicaciones que van desde sistemas en tiempo real o transaccionales a sistemas distribuidos multimedia; por otro lado el XTP es capaz de operar sobre IP (lo cual hace que sea usable en un entorno Internet), sobre CLNP (lo cual cubre las redes OSI), sobre los subniveles

[4] La Universidad Politécnica de Madrid forma parte del *XTP Forum*.

[5] La primera versión del XTP apareció en 1987 y fue propuesta conjuntamente por *Protocol Engine Inc.* y el *XTP Forum*. La versión actual es la 4.0 (Marzo 1995) y de una de las novedades de esta versión ha sido el cambio de nombre, anteriormente éste era *Xpress Transfer Protocol*.

LLC o MAC de cualquier RAL, y directamente sobre el nivel de adaptación de una red ATM.

El XTP proporciona, en un único protocolo de transporte, servicios de distribución punto a punto y multipunto; y a su vez para cada uno de estos servicios hay disponibles un amplio rango de funcionalidades seleccionables por el usuario del servicio. A continuación se exponen las principales características del XTP asociadas al servicio de distribución multipunto:

- *Paradigma de la conexión.* El XTP sigue un paradigma multipunto orientado a conexión, en el cual cada transmisor establece una conexión punto-multipunto simplex (1→N) con un conjunto de receptores.

- *Algoritmos de control.* XTP proporciona a los usuarios la posibilidad de activar o desactivar para cada conexión los procedimientos de control de errores y de flujo. Si los algoritmos de control están activados, el principal aspecto que afecta a la robustez del protocolo es cómo el transmisor conoce y procesa la información de estado de todos los receptores. La comunidad XTP ha experimentado y propuesto dos métodos para los algoritmos de control: (1) un algoritmo heurístico basado en temporizadores (denominado el *algoritmo de bucket*), (2) un procesamiento explícito de la información de estado de cada receptor en el grupo (llamado *multipunto basado en listas*). Cuando se usa el algoritmo de bucket, el transmisor solicita información de control a los receptores y espera un tiempo definido por la aplicación antes de procesar los paquetes de control que ha recibido. Los paquetes de control son procesados sin determinar qué receptores han enviado dichos paquetes y no hay un conocimiento explícito del conjunto de receptores por parte del transmisor. De este modo la fiabilidad queda muy limitada, reduciéndose a que si la transmisión se completa, cada paquete de datos ha sido recibido por al menos un receptor. La filosofía del *algoritmo de bucket* es lograr un esquema que escale bien ante grupos con muchos receptores, evitando la necesidad de un conocimiento explícito del conjunto de receptores por parte del transmisor. Sin embargo, la experiencia ha mostrado [DLW94] que resulta difícil ajustar correctamente el mencionado temporizador, y que el grado de fiabilidad obtenido no es aceptable para muchas aplicaciones. Esta experiencia llevo a incorporar al XTP un *multipunto basado en listas*. En este modo, el transmisor mantiene una tabla de todos los receptores activos y cada vez que solicita información de control, espera a recibir una contestación por cada uno de los receptores; consecuentemente, es posible proporcionar un servicio más fiable. La versión 4.0 del XTP soporta únicamente el *multipunto basado en listas*, mientras que la versión anterior 3.6 también incorporaba *el algoritmo de bucket.*

- ❒ *Asentimientos selectivos.* XTP permite al usuario decidir si deben enviarse asentimientos y cuando. Los receptores envían asentimientos únicamente cuando el transmisor lo solicita, permitiendo al usuario seleccionar una frecuencia de asentimientos que puede variar desde siempre, a algunas veces, o a nunca. Filosóficamente la generación de asentimientos por parte de los receptores está desacoplada con la llegada de datos y el tamaño de la ventana. Cuando el transmisor desea conocer el estado de la conexión, lo pregunta y los receptores le responden.

- ❒ *Control de errores seleccionable.* Soporta distintos tipos de control de errores seleccionables por el usuario del servicio: (a) modo totalmente fiable, proporciona un servicio de fiabilidad tipo TCP o TP4; (b) servicio "mejor esfuerzo"[6], proporciona un servicio de fiabilidad tipo UDP en el cual el receptor no envía asentimientos positivos y por tanto el receptor nunca sabe si los datos fueron distribuidos con éxito; (c) modo *"fastnak"*, cuando se usa esta opción, un receptor al identificar datos fuera de secuencia, envía inmediatamente un paquete de control que informa al transmisor sobre los datos que le faltan, seguidamente éste reenvía los datos solicitados; (d) modo *"noerror"*, en este modo se suspende el esquema de retransmisiones, los datos recibidos correctamente se entregan en secuencia, pero si falta algún paquete intermedio no se retransmite (útil para enviar voz y vídeo digital).

- ❒ *Control de flujo seleccionable.* Soporta distintos tipos de control de flujo que son seleccionables por el usuario del servicio: (a) basado en ventana deslizante, (b) modo reserva, es una política de control de flujo conservadora que asegura que los datos nunca serán descartados por falta de buffers en el destino, (c) sin control de flujo.

- ❒ *Esquema de retransmisiones.* Permite que el usuario del servicio elija entre un esquema de retransmisión simple[7] y retransmisiones selectivas.

- ❒ *Control de pertenencia al grupo.* XTP incluye mecanismos que permiten a los receptores unirse al grupo, antes de que el transmisor establezca la conexión multipunto o posteriormente cuando la comunicación ya está en curso, y dejar el grupo. El XTP no impone políticas para gestionar el grupo de receptores, ya que éstas vienen marcadas por la aplicación; consecuentemente, XTP informa a la aplicación que está transmitiendo, de los cambios en la pertenencia y le permite decidir qué debe hacerse. Ya que el XTP 4.0 sigue la filosofía de *multipunto basado en listas,* el

[6] *Best-effort* en terminología anglosajona.

[7] También conocido como *"go-back-n"*.

transmisor debe conocer en cada momento quiénes son todos los receptores, y de aquí que todos los cambios en la pertenencia le sean comunicados al transmisor.

Las principales críticas a la versión actual del XTP son: (1) cuando el transmisor solicita información de control, cada uno de los receptores debe contestar en modo punto a punto, y aunque XTP incluye un mecanismo que retarda de forma aleatoria estas contestaciones con el fin de evitar que sean simultáneas, este comportamiento impide que el XTP escale bien con respecto al tamaño del grupo, (2) las retransmisiones son realizadas desde la fuente y son enviadas en modo multipunto (aunque únicamente un receptor haya solicitado dicho paquete), esto hace que el XTP no optimice el uso de los recursos de red.

Una versión mejorada del XTP está siendo utilizada como protocolo de transporte multipunto en un proyecto Europeo Race (R2060) llamado CIO (*Coordination, Implementation and Operation of Multimedia Services*).

El objetivo del proyecto CIO[8] es el desarrollo de teleservicios multimedia en sistemas finales multivendedor, así como proporcionar la correspondiente plataforma de comunicaciones. Como teleservicios multimedia han desarrollado el MMMS (*Multimedia Mail Service*), una herramienta para enviar y recibir mensajes de contenido arbitrario (texto, imágenes, audio y vídeo), y JVTOS (*Joint Viewing and Teleoperation Service*) herramienta que soporta grupos de trabajo remotos en combinación con una conferencia audio/vídeo. El protocolo de transporte propuesto ha sido el XTPX (*Xpress Transfer Protocol eXtended*), protocolo derivado del XTP 3.6 que incorpora un QoS más flexible y, permite una negociación y renegociación de parámetros.

2.2.3 RMP

El RMP (*Reliable Multicast Protocol*) [Mon94, Whe95][9] es un protocolo desarrollado en la Universidad de West Virginia en cooperación con la NASA, y que próximamente será presentado a consideración a la *"Internet Society"* para obtener el grado de *"Standard Internet"*. El RMP está basado en un protocolo propuesto por Chang y Maxemchuk [CM84].

[8] Información sobre el proyecto CIO se encuentra disponible en *http://www.prz.tu-berlin.de/docs/html/CIO/cio.engl.html.*

[9] Información adicional sobre el RMP se encuentra disponible en *http://research.ivv.nasa.gov/projects/RMP.html.*

El trabajo de Chang y Maxemchuk fue uno de los pioneros en multipunto fiable y consiste básicamente en un esquema distribuido, en el que todos los procesos juegan el mismo papel en la comunicación, que proporciona una distribución fiable y totalmente ordenada a los miembros del grupo. Dichos miembros están ordenados en un anillo lógico, siendo uno de ellos el receptor primario (denominado *token site*). La responsabilidad de receptor primario va desplazándose por el anillo de miembro en miembro después de cada transmisión de datos. El receptor primario tiene principalmente dos funciones: (1) asentir a la fuente la recepción de los nuevos mensajes enviados por ésta (mediante un ACK multipunto), y (2) retransmitir al resto de los receptores los mensajes que éstos soliciten (solicitudes de retransmisión que se hacen usando paquetes NAK). Los asentimientos positivos enviados por el receptor primario incluyen un número de secuencia especial (denominado *timestamp*), que permite conseguir la ordenación total de todos los mensajes enviados al grupo, aunque éstos provengan de fuentes distintas; adicionalmente, dichos asentimientos positivos sirven para pasar la responsabilidad de receptor primario al siguiente miembro en el anillo. Este esquema plantea principalmente dos inconvenientes: (1) dado que el receptor primario es el responsable de procesar todos los asentimientos negativos enviados por el resto de miembros del grupo y de realizar las correspondientes retransmisiones, este punto se convierte en un cuello de botella cuando hay fallos; (2) cada vez que un miembro falla, o simplemente hay una entrada/salida en el grupo, es necesario reformar el anillo lógico, lo cual impide que esta solución escale bien con respecto al tamaño del grupo.

El RMP es un protocolo de transporte que proporciona un servicio multipunto fiable, atómico y totalmente ordenado, sobre un servicio IP multipunto [Dee89], para ello sigue básicamente la propuesta de Chang y Maxemchuk aunque incorpora una serie de mejoras. Entre éstas las más significativas son las siguientes:

- *Se ha ampliado el rango de niveles de QoS disponibles.* Una aplicación RMP puede elegir para cada mensaje enviado un nivel de QoS, variando éstos desde no fiable y no ordenado, a totalmente ordenado y totalmente resistente a fallos (fiable) [Mon94]. A mayor nivel de QoS, mayor será el retardo de distribución de cada mensaje.

- *El algoritmo de reforma del anillo lógico se ha optimizado.* En RMP también es necesario ejecutar un proceso de reforma cada vez que un miembro falla o se produce una partición en la interred, pero el algoritmo ha sido optimizado. Cada vez que se ejecuta un proceso de reforma, el nivel RMP notifica el hecho a la aplicación, y de acuerdo con estas notificaciones la aplicación decide, según su propia semántica, que acciones deben tomarse.

- *Cambios en la pertenencia menos costosos.* Las entradas y salidas de miembros en el grupo no obligan a ejecutar un proceso de reforma del anillo lógico, y cada vez que se produce un cambio en el grupo, el nivel RMP notifica a la aplicación el hecho, informando de la clase de operación que ha ocurrido (entrada o salida) y del estado del grupo después del cambio.

- *Asentimientos acumulativos.* RMP permite que múltiples mensajes sean asentidos con un único ACK.

- *Traducción de nombres a direcciones.* La traducción de los nombres de grupo a direcciones IP multipunto puede ser manejado por una autoridad externa o por el propio RMP.

Las principales críticas al RMP son las siguientes: (1) al igual que en la propuesta de Chang y Maxemchuk el receptor primario se puede convertir en un cuello de botella cuando hay fallos; (2) dado que todos los miembros deben conocer en todo momento quiénes forman el anillo lógico, cada vez que se produce un cambio en la pertenencia es necesario distribuir a todos los miembros la nueva composición del anillo lógico, este comportamiento hace que el protocolo no sea adecuado para grupos con muchos miembros y muy dinámicos.

2.2.4 MTP y MTP-2

El MTP (*Multicast Transport Protocol*) [AFM92] es el primer protocolo multipunto fiable que fue publicado por la *"Internet Society"* como RFC, actualmente éste tiene el estado de *"Informational"*.

El MTP es un esquema centralizado que proporciona un servicio de distribución multipunto fiable y totalmente ordenado. En MTP se distinguen diferentes papeles entre los participantes de una conexión multipunto: (1) *el maestro,* primer miembro que se unió al grupo, es responsable de mantener y distribuir los testigos (que aseguran el orden total), y de gestionar el grupo (autoriza que nuevos miembros se unan o abandonen la conexión MTP); (2) *los productores*, sistemas finales que envían datos, antes de enviar cada mensaje (uno o más paquetes), deben obtener un testigo desde el maestro, este testigo incluye un número de secuencia que permite lograr el orden total; (3) *los consumidores*, reciben los datos y usan asentimientos negativos para solicitar retransmisiones. Estos asentimientos negativos son enviados en modo punto a punto al productor correspondiente, el cual retransmite en modo multipunto los datos solicitados al grupo completo.

Existen tres parámetros fundamentales que determinan la operación y grado de fiabilidad del protocolo:

- *"Heartbeat"*: es un intervalo de tiempo.

- *Ventana*: define el número máximo de paquetes que cada productor puede enviar al grupo en cada *"heartbeat"*.

- *Retención*: entre otras cosas define el periodo de tiempo (en *"heartbeats"*), que un productor debe tener almacenados los datos, por si acaso un consumidor solicita su retransmisión.

Un estudio del rendimiento del MTP, en base a una serie de simulaciones puede encontrarse en [SH95].

El MTP-2 protocolo ha sido diseñado e implementado por un grupo de investigadores alemanes de las Universidades de Berlín y Bremen. Componentes de este grupo han realizado distintas contribuciones en el área de la comunicación multipunto, siendo una de las más significativas una herramienta para las aplicaciones de conferencia multimedia que permite compartir una ventana X, herramienta llamada X_Y. Inicialmente X_Y fue implementada sobre el MTP y como consecuencia de la experiencia obtenida diseñaron un sucesor para el MTP, llamado MTP-2 (*Multicast Transport Protocol version 2*) [BOG+94].

El MTP-2 ha sido propuesto [OB94] para ser usado como protocolo de transporte multipunto, que proporcione el servicio de comunicación multipunto para aplicaciones de conferencia multimedia definido por el ITU-T[10] [ITU93].

El MTP-2 incorpora una serie de mejoras al MTP en los aspectos de escalabilidad, eficiencia, fiabilidad y robustez. Las aportaciones más significativas son las siguientes:

- *Recuperación ante la pérdida del maestro.* En MTP el maestro es un punto crítico, de modo que un fallo de dicho maestro o una partición en la red pueden impedir que la comunicación continúe. Con el fin de proporcionar más robustez, el MTP-2 incorpora un mecanismo que permite la recuperación ante la pérdida del maestro.

- *Mecanismo para el cambio de maestro.* En determinadas situaciones tales como, la sobrecarga del miembro que hace funciones de maestro, el deseo de dicho miembro

[10] Antes conocido como CCITT.

de salir del grupo, o conexiones donde existe un único productor y sería más óptimo que éste realizara las funciones de maestro para evitar que los testigos tuvieran que viajar por la red, es útil disponer de un mecanismo que permita el cambio de maestro después de iniciada una conexión; de aquí que esta funcionalidad haya sido incorporada por el MTP-2.

- *Ajuste dinámico de parámetros.* En determinadas situaciones puede ser útil ajustar dinámicamente, después de iniciada una conexión, los parámetros de *"heartbeat"*, *ventana* y *retención*; por ejemplo, con el fin de evitar tráfico innecesario (en cada *"heartbeat"* se envía al menos un paquete, y en el caso de que no haya ningún productor activo el maestro envía paquetes vacíos), o para aumentar la fiabilidad (valores más altos de *"heartbeat"* y *retención* incrementan la probabilidad de una retransmisión con éxito, al almacenarse los paquetes durante más tiempo). El MTP-2 incluye un mecanismo que permite que el maestro ajuste dinámicamente estos parámetros.

- *Mejora del procedimiento de unión grupo.* En MTP un nuevo miembro no puede unirse al grupo en cualquier momento, restricción que ha sido eliminada en el MTP-2.

Las principales críticas al MTP y MTP-2 son: (1) proporciona un grado de fiabilidad bastante limitado, ya que los productores nunca están totalmente seguros de que un mensaje fue recibido; (2) debido a la incertidumbre de distribución, los productores deben mantener los paquetes enviados un periodo de tiempo considerado suficiente como para proporcionar una garantía razonable de que todos los consumidores han recibido el paquete; esto obliga a un gran espacio de buffer o a un bajo caudal; y (3) a pesar de incluir mecanismos que permiten la recuperación ante la pérdida del maestro, este sistema siempre constituirá un punto crítico al ser un posible cuello de botella que puede limitar el rendimiento del protocolo.

2.2.5 Otros protocolos

El número de protocolos de multipunto fiable que han ido apareciendo en los últimos años es considerable [MTP]. En general, estas propuestas han sido desarrolladas de forma paralela al protocolo planteado en este trabajo, y aún no están muy perfiladas, siendo trabajos en curso. Sin embargo, no están exentas de interés al apuntar ideas novedosas que en

algunos casos coinciden parcialmente con lo aquí planteado. De entre todos estos protocolos se han elegido por su representatividad los siguientes:

- **LBRM** (*Log-Based Receiver_reliable Multicast*) [HSC95]. Este protocolo opera a nivel de transporte y ha sido diseñado para soportar aplicaciones de Simulación Interactiva Distribuida. Sigue una aproximación orientada al receptor y las retransmisiones se realizan desde servidores de "log" de dos categorías (primario o secundario). Los miembros solicitan las retransmisiones a los servidores secundarios, los cuales a su vez, lo hacen al servidor de primario. Los sistemas retransmisores utilizan esquemas probabilísticos para decidir si las retransmisiones se harán en modo punto a punto o multipunto. Para que los miembros detecten lo antes posible las posibles pérdidas, la fuente distribuye cada cierto tiempo (periodo variable) un "mensaje de sesión" indicando cual ha sido el último paquete enviado al grupo.

- **TMTP** (*Tree-based Multicast Transport Protocol*) [YGS95]. Este protocolo opera a nivel de transporte y sigue esencialmente una aproximación orientada al receptor aunque adicionalmente utiliza ACKs periódicos. El TMTP agrupa a los miembros en dominios jerárquicos eligiendo un gestor de dominio, asimismo organiza los gestores de dominio en un árbol de control. Dicho árbol de control se utiliza para realizar las funciones de control de flujo y control de errores. Cada gestor de dominio es responsable de realizar retransmisiones a los miembros de su dominio y a los dominios vecinos. El TMTP opera sobre el IP multipunto y para limitar el alcance de las retransmisiones utiliza el campo TTL[11]. Como mecanismo de control de flujo utiliza una combinación de control de caudal, cada fuente tiene asignada de forma fija un caudal máximo, y un mecanismo de pseudoventanas deslizantes estáticas gestionadas por los sistemas retransmisores.

- **AFDP** (*Adaptive File Distribution Protocol*) [CK96]. En este protocolo cada vez que una fuente (editor) desea enviar información a un grupo debe solicitar previamente permiso a un sistema con responsabilidades especiales denominado "secretaria". La "secretaria" tiene como principales funciones hacer un control del caudal enviado al grupo y realizar tareas de gestión de grupo. Para evitar una implosión de NAKs, los miembros (suscriptores) después de detectar un salto en la secuencia retardan un tiempo aleatorio el envío del NAK correspondiente; dichos NAKs se envían utilizando TCP. Las retransmisiones son realizadas desde la fuente en modo multipunto.

[11] *Time To Live.*

- **FTM** (*Fault-Tolerance internet Multicasting*) [Raj91]. Este protocolo opera a nivel de red sobre un protocolo de encaminamiento multipunto propuesto por el mismo autor y sigue una aproximación orientada al emisor. El FTM opera en ciclos, de modo que al final de cada ciclo se realiza un "checkpoint" que sirve para decidir que paquetes deben ser retransmitidos y cuales han sido recibidos por todos los miembros, y consecuentemente pueden ser liberados los buffers asociados. El FTM utiliza un árbol de control para realizar una agregación de los mensajes de control asociados a cada "checkpoint". Las retransmisiones se hacen en modo multipunto a todo el grupo y son realizadas siempre desde la fuente.

- **RMTP** (*Reliable Multicast Transport Protocol*) [LS96]. Este protocolo opera a nivel de transporte. El conjunto de receptores en una misma región local son agrupados y se les asigna un "receptor designado" que se encarga de descargar de trabajo a la fuente procesando los ACKs enviados por los receptores de dicha región y si procede realizando las correspondientes retransmisiones. Los receptores del grupo envían periódicamente ACKs informando de qué paquetes han recibido y cuales necesitan ser retransmitidos. Si el número de miembros que solicita un determinado paquete está por encima de un determinado umbral la retransmisión se hace en modo multipunto y en caso contrario se hace en modo punto a punto. Como mecanismos de control de flujo limita el caudal de datos enviado al grupo y utiliza un sistema de ventana deslizante en la fuente.

2.3 Multipunto a nivel de red

Cada día son más numerosas las redes que ofrecen un servicio multipunto. La mayoría de las RALs y más recientemente, tecnologías para redes de área extensa tales como SMDS [LP91] y ATM [Pry91] especifican el multipunto como parte de su servicio. Adicionalmente, este servicio es ofrecido en interredes basadas en datagramas siendo el modelo IP multipunto, desarrollado por Deering [Dee89, Dee91], el más representativo.

En las siguientes secciones se expone en primer lugar, el modelo de servicio multipunto para interredes basadas en datagramas planteado por Deering y posteriormente, las principales propuestas al problema del encaminamiento multipunto para este tipo de interredes.

2.3.1 Modelo de Deering

El modelo de servicio multipunto para interredes basadas en datagramas, planteado por Deering [Dee91], llamado **Modelo de Grupo de Sistema Final** (*Host group model*), es el modelo de servicio multipunto más ampliamente aceptado.

Los principales objetivos que se planteó Deering al desarrollar este modelo es que fuera flexible y genérico. Las propiedades del *Modelo de Grupo de Sistema Final* son las siguientes:

- El conjunto de destinos de un paquete multipunto es denominado *grupo* y es identificado por una dirección de grupo o dirección multipunto. Esta propiedad hace posible el *direccionamiento lógico*, y esto significa que la fuente puede mandar datos a un grupo sin conocer a priori quiénes son los miembros de dicho grupo. Debe observarse que esta cualidad puede ser útil para muchas aplicaciones, tales como servicios de distribución de noticias, videoconferencia, etc.

- Las fuentes de un grupo no necesitan ser miembros de dicho grupo. Este modelo de grupo es llamado *grupo abierto,* en contraposición a los *grupos cerrados* donde se obliga a que las fuentes sean al mismo tiempo miembros del grupo.

- Los grupos pueden tener cualquier número de miembros.

- Los miembros pueden tener cualquier localización, esto significa, que no hay ninguna restricción topológica a la localización de los miembros.

- La pertenencia a un grupo es dinámica y autónoma. Los miembros pueden unirse o dejar un grupo en cualquier momento, y no necesitan sincronizarse o negociar con otros miembros del grupo o con las potenciales fuentes. Este modelo de grupo es denominado *dinámico,* frente a los *grupos estáticos* que no permiten que los miembros entren y salgan en cualquier momento.

- Los grupos pueden ser *permanentes o temporales.* Un grupo permanente es aquel que tiene asignado de forma permanente una dirección de grupo, asignación que habrá sido realizada por una autoridad administrativa. Un grupo permanente existe siempre y puede tener cualquier número de miembros incluido cero. Por el contrario, los grupos temporales sólo existen en tanto en cuanto tengan al menos un miembro y la dirección de grupo les es asignada de forma temporal.

- Es posible limitar el alcance de una transmisión multipunto a un número de saltos, o a una determinada área (por ejemplo un dominio administrativo).

- El grado de fiabilidad ofrecido es el de "mejor esfuerzo". Esto significa que el servicio no garantiza que los paquetes no se pierdan, lleguen dañados, duplicados o fuera de orden.

Estas propiedades permiten que este modelo de servicio multipunto sea muy genérico y como consecuencia útil para un espectro muy amplio de aplicaciones multipunto, factor que ha resultado clave para su mayoritaria aceptación.

2.3.2 Estrategias de encaminamiento multipunto

El encaminamiento multipunto en interredes operando en modo datagrama se basa principalmente en la utilización de *árboles de distribución*. Estos árboles son construidos a partir del grafo definido por la interred, considerando los nodos de la interred (encaminadores y sistemas finales) como vértices del grafo y las facilidades de comunicación entre estos nodos (enlaces y subredes) como aristas del grafo. El árbol de distribución usado para distribuir datos al grupo g (T_g) es calculado, a partir del grafo definido por la interred, utilizando un algoritmo de encaminamiento multipunto y una función coste. Este árbol T_g tiene la particularidad de contener a todos los miembros del grupo g, de modo que al propagar cada paquete enviado al grupo g por todo el árbol T_g se garantiza que cada uno de los miembros del grupo obtiene una copia del paquete.

Los árboles de distribución pueden clasificarse en árboles compartidos o centrados y, árboles fuente. Cuando se utilizan árboles compartidos existe un único árbol por grupo que es compartido por todas las fuentes que envían datos a dicho grupo. Mientras que cuando se utilizan árboles fuente existe un árbol de distribución por cada par (grupo, fuente). De modo que cuando una fuente f envía un paquete al grupo g éste se distribuye por el árbol $T_{g,f}$, siendo la fuente la raíz de dicho árbol.

Los primeros esquemas de encaminamiento multipunto propuestos utilizaban árboles fuente mientras que los trabajos más recientes en este área, plantean la utilización de árboles compartidos o soluciones híbridas. En las siguientes secciones se exponen en más detalle las principales propuestas basadas en árboles fuente así como, las actuales tendencias en el área del encaminamiento multipunto.

Existe abundante literatura sobre la bondad de los distintos algoritmos de construcción de árboles de distribución, aspecto que se sale fuera del ámbito de este trabajo, pero sin embargo, al lector interesado en este tema se le recomiendan los siguientes trabajos [Win87, Wax88, SBK94, WE94, Sab95, SL95, Wei95].

2.3.2.1 Principales algoritmos basados en árboles fuente

Los algoritmos de encaminamiento multipunto y protocolos asociados que se van a exponer, son extensiones a los algoritmos de encaminamiento punto a punto y protocolos asociados más utilizados, que aprovechan las estructuras de datos ya presentes y el servicio ofrecido por estos protocolos punto a punto. El **DVMRP** (*Distance-Vector Multcast Routing Protocol*) [WPD88] es una extensión al protocolo vector-distancia RIP (*Routing Information Protocol*) [Hed88], mientras que el **MOSPF** (*Multicast Open Shortest-Path First*) [Moy94a] es una extensión al protocolo de estado del enlace OSPF (*Open Shortest-Path First*) [Moy89].

Estos protocolos de encaminamiento multipunto, principalmente el DVMRP[12], están siendo utilizados actualmente en la MBone.

2.3.2.1.1 DVMRP

El DVMRP [WPD88] es un protocolo que implementa el algoritmo multipunto vector-distancia propuesto por Deering [Dee91]. Este algoritmo usa vectores (destino, distancia) para anunciar las subredes que tienen funcionalidad multipunto y la distancia hasta éstas. Dichos vectores son intercambiados entre encaminadores vecinos que disponen de funcionalidad multipunto.

El DVMRP está basado principalmente en un algoritmo, para realizar difusión en interredes, de Dalal y Metcalfe [DM78] llamado RPF (*Reverse Path Forwarding*). Deering modificó el RPF para eliminar la posibilidad de que varias copias de un mismo paquete fueran propagadas por el mismo enlace multiacceso [DC90]. Adicionalmente, incorporó un mecanismo especial llamado *difusión truncado* que evita la retransmisión de los paquetes

[12] La implementación concreta del DVMRP que se está utilizando en la MBONE varía en algunos detalles de la especificación del protocolo realizada en [WPD88].

con destino al grupo *g* a las subredes hoja[13] que no tienen conectado a ningún miembro de *g*, como resultado se obtiene el llamado árbol de difusión truncado.

El funcionamiento del algoritmo es el siguiente: cuando una fuente comienza a mandar paquetes a un grupo, éstos se propagan a través del árbol de difusión truncado. En esta situación, cuando un encaminador conectado a una subred hoja, recibe un paquete de datos dirigido a un grupo y descubre, que no hay miembros de este grupo en dicha subred, envía un mensaje de "poda" a su predecesor en el árbol. Si un encaminador recibe mensajes de poda de todos sus encaminadores subordinados y además sus propias subredes hoja tampoco tienen miembros del grupo, él mismo enviará un mensaje de poda a su predecesor. De esta forma, la información sobre la ausencia de miembros del grupo se va propagando hacia arriba en el árbol, por las ramas que no conducen a miembros, dando lugar a un árbol de caminos mínimos con raíz en la fuente, en el cual todas las subredes hoja tienen miembros del grupo, siendo éste un verdadero árbol de distribución multipunto. Paquetes posteriores enviados por la misma fuente al mismo grupo no atravesarán las ramas que han sido podadas ni los encaminadores que conducen a dichas ramas. Al cabo de un cierto tiempo, las ramas podadas se incorporan otra vez al árbol; realizándose nuevamente la poda si la fuente continúa enviando paquetes y todavía no hay miembros del grupo en dichas ramas.

Si en un momento dado, en una rama que está podada aparece un miembro, el encaminador que detecta el hecho envía un mensaje de "injerto" solicitando que dicha rama sea de nuevo incorporada al árbol.

El DVMRP permite a los encaminadores multipunto construir un árbol de distribución con raíz en la fuente y con caminos mínimos hasta los miembros del grupo.

2.3.2.1.2 MOSPF

El MOSPF [Moy94a, Moy94b, Moy94c] es un protocolo que implementa el algoritmo multipunto de estado del enlace propuesto por Deering [Dee91]. El encaminamiento del tipo estado del enlace precisa que los encaminadores periódicamente monitoricen el estado de todos (o parte) de sus enlaces adyacentes. Esta información de estado es posteriormente comunicada al resto de encaminadores utilizando un protocolo de inundación de propósito especial. Con el fin de añadir funcionalidad multipunto a un algoritmo del tipo esta-

[13] Una subred hoja en el árbol definido para una fuente y un grupo, es aquella que no es utilizada por ningún encaminador para enviar datos a dicha fuente. Las subredes con un único encaminador siempre son subredes hoja.

do del enlace, la información sobre la presencia o no de miembros en un enlace es considerada como parte de la "información de estado" de dicho enlace (información denominada *de pertenencia a los grupos*). Como consecuencia, cada vez que un enlace empieza a formar parte de un grupo (un primer sistema final conectado al enlace se suscribe al grupo), o deja de formar parte de un grupo (un último sistema final conectado al enlace deja el grupo), cambia la información de estado del enlace, hecho que origina, que el encaminador que se encarga de monitorizar el estado de dicho enlace (encaminador designado) informe *al resto de los encaminadores de la red del cambio de estado. De este modo, todos los* encaminadores tienen información sobre dónde están localizados los miembros de cada grupo, y utilizando esta información y el algoritmo de Dijkstra, pueden calcular el árbol de caminos más cortos desde cada fuente a cada grupo.

Los encaminadores calculan cada uno de los árboles asociados a un par (fuente, grupo) cuando reciben un primer paquete enviado por dicha fuente y con destino a dicho grupo. Esta aproximación relativa a cuando calcular cada árbol de distribución, se denomina *construcción bajo demanda.*

2.3.2.1.3 Principales críticas al DVMRP y MOSPF

Las dos principales críticas realizadas a los algoritmos vistos son las siguientes:

- **Escalabilidad.** Los algoritmos de encaminamiento basados en árboles fuente no escalan bien con el tamaño de la interred y con el número de fuentes. Estos algoritmos almacenan información por cada una de las fuentes, y por tanto, si S es el número de fuentes activas de un grupo multipunto y N es el número de grupos multipunto presentes, nos da un factor de escalabilidad de SxN, característica que limita su aplicabilidad cuando, para un número de grupos dado, aumenta el número de fuentes.

  En MOSPF los encaminadores tendrán que guardar la información de pertenencia a un grupo aunque en sus subredes directamente conectadas, no haya ningún miembro del grupo, ni éstas pertenezcan a ningún árbol multipunto para ese grupo. En DVMRP cada encaminador, aunque no esté interesado en enviar/recibir paquetes multipunto a/desde un grupo, debe encargarse de la recepción, generación e interpretación de los mensajes de poda, y de mantener la información relativa a estas podas por cada par (fuente, grupo). Estos comportamientos limitan la escalabilidad con el tamaño de la interred.

Otro aspecto a considerar que también limita la escalabilidad con el tamaño de la interred es el hecho de que ambos algoritmos se comportan a veces, como mecanismos de difusión. En MOSPF se hace difusión de la información de pertenencia a los grupos, esto implica que cada vez que haya un cambio de pertenencia en cualquiera de los grupos se producirá un difusión a todo el dominio. En DVMRP inicialmente los paquetes de datos se envían a todos los encaminadores, ya que la pertenencia a un grupo es asumida hasta que los encaminadores sin miembros locales se encarguen de mandar explícitamente los mensajes de poda, para ser eliminados del árbol de distribución. En ambos casos, todos los encaminadores tendrán una sobrecarga de procesamiento por cada grupo existente en la interred.

- **Dependencia del algoritmo de encaminamiento punto a punto.** Los algoritmos planteados tienen una gran dependencia del algoritmo de encaminamiento punto a punto empleado. Cuando se utiliza MOSPF el algoritmo de encaminamiento punto a punto empleado debe ser OSPF, y cuando se utiliza DVMRP el algoritmo de encaminamiento punto a punto empleado debe ser RIP. Este fuerte acoplamiento entre los algoritmos multipunto y los algoritmos punto a punto, dificulta la evolución de estos protocolos y trae consigo soluciones poco flexibles.

2.3.2.2 Actuales tendencias

Recientemente han aparecido nuevas propuestas al problema del encaminamiento multipunto que tratan de resolver los problemas de los algoritmos antes expuestos. A continuación se comentan brevemente las dos nuevas arquitecturas de encaminamiento multipunto que están teniendo más aceptación por parte de los investigadores del área.

2.3.2.2.1 CBT

La arquitectura CBT (*Core Based Tree*) [BFC93, Bal96, BRJ96] utiliza árboles compartidos. Esto significa que existe un único árbol de distribución por grupo que es compartido por todas las fuentes del grupo; este árbol compartido tiene como raíz un encaminador llamado "*core*". La idea central de esta arquitectura de encaminamiento es, que usando un único árbol compartido por grupo, aunque no se minimizan los retardos sufridos por los paquetes hasta que alcanzan a los miembros del grupo, pueden obtenerse valo-

res cercanos al retardo mínimo y al mismo tiempo se asegura que no haya problemas de escalabilidad. Las principales características del CBT son las siguientes:

- ❒ Es totalmente independiente del algoritmo de encaminamiento punto a punto.
- ❒ En ninguna circunstancia se comporta como un mecanismo difusión.
- ❒ La filosofía de construcción del árbol de distribución es diferente a la vista anteriormente. Un encaminador no forma parte de un árbol de distribución a menos que envíe explícitamente un mensaje solicitando unirse al árbol. Un encaminador solicita pertenecer al árbol asociado a un grupo cuando dicho encaminador está en el camino entre al menos un miembro de dicho grupo y el *"core"*.

Las principales críticas al CBT se derivan del hecho de usar árboles compartidos y son:

- ❒ Al usar un árbol compartido por todas las fuentes, se produce una *concentración de tráfico* en los enlaces y encaminadores que pertenecen a dicho árbol.
- ❒ *Los paquetes no se propagan por los caminos más cortos* desde la fuente a cada miembro y como consecuencia, esta estrategia de encaminamiento no es adecuada para aplicaciones sensibles al retardo [SL95].

2.3.2.2.2 PIM

Cada tipo de árboles, fuente y compartido, tiene sus ventajas e inconvenientes y dependiendo de las características del grupo y de las necesidades de la aplicación será más apropiado un tipo de árbol u otro. Esta idea básica ha llevado a Deering y a otros a plantear la arquitectura de encaminamiento multipunto denominada PIM (*Protocol Independent Multicast*) [DEF+94, DEF+96, EFJ+96]. PIM es capaz de trabajar con árboles fuente y con árboles compartidos y como resultado tiene dos modos básicos de funcionamiento: modo disperso (*sparcely*) y modo denso (*dense*).

El PIM **modo denso** [EFJ+96] utiliza árboles fuente y es bastante similar al DVMRP pero sin depender de ningún protocolo de encaminamiento punto a punto. En este modo, al igual que en DVMRP, se utilizan paquetes de poda e injerto. El modo denso está pensado para ser usado en entornos ricos en recursos tales como, redes de campus formadas por un conjunto de RALs interconectadas.

El PIM **modo disperso** [DEF+96] será normalmente adecuado para ser usado en interredes muy extensas. Cuando se utiliza este modo, los miembros de un determinado grupo inicialmente reciben los datos multipunto a través de un árbol, compartido por todas las fuentes del grupo, árbol que se va formando cuando los miembros indican explícitamente su deseo de unirse a dicho árbol. Posteriormente, uno o más miembros, con el fin de mejorar el retardo de distribución, pueden solicitar que desean recibir los datos provenientes de una fuente en concreto, a través de caminos mínimos, este deseo se expresa enviando un mensaje hacia dicha fuente que permite que vaya creándose un árbol con raíz en ésta. Cuando un miembro crea el camino más corto hacia una fuente particular, él mismo se poda del árbol compartido para ese par (fuente, grupo), sin embargo continuará recibiendo los datos enviados al grupo por el resto de las fuentes a través del árbol compartido.

El árbol compartido se va construyendo alrededor de unos encaminadores llamados RPs (*Rendez-vous Points*) que tienen como finalidad hacer el protocolo más robusto. Si existen varios RPs cada uno de ellos será la raíz de un árbol compartido. Un nuevo miembro sólo necesita unirse a un RP, si es que hay varios, pero sin embargo el emisor debe enviar cada paquete de datos a cada uno de los RPs, de modo que todos los miembros del grupo recibirán una copia del paquete.

El PIM es un trabajo en progreso y todavía tiene determinados aspectos sin resolver totalmente, como por ejemplo, ¿cuándo y cómo se conmuta de un árbol compartido a un árbol fuente?. La principal crítica realizada a PIM es su complejidad [SL95], algunos autores ya se han planteado si las mejoras obtenidas al conmutar del árbol fuente al árbol compartido son lo suficientemente significativas como para justificar la complejidad del protocolo [Bal95].

Capítulo 3.

PLANTEAMIENTO DEL PROBLEMA

Con la presente tesis se pretende hacer una contribución al problema de la fiabilidad multipunto en interredes extensas basadas en datagramas. Con este fin, en este capítulo se analiza cual ha sido el enfoque tradicional, identificando las posibles deficiencias, para posteriormente hacer un replanteamiento del problema de acuerdo con una serie de objetivos planteados.

La aproximación tradicional al problema de la fiabilidad multipunto se ha basado principalmente en los dos siguientes puntos:

- Proporcionar en un mismo protocolo servicios de fiabilidad, atomicidad y ordenación en distintos grados (orden total, orden causal, etc.). Sin embargo, la realidad es que muchas aplicaciones multipunto que precisan fiabilidad no necesitan otros servicios, tales como atomicidad, orden total o causal, y consecuentemente, la implementación de estas aplicaciones sobre los mencionados protocolos conduce a soluciones altamente ineficientes en tiempo de ejecución [RL93] (por ejemplo, se retarda de forma innecesaria la entrega de paquetes o se utilizan muchos buffers).

- Extender al entorno multipunto las soluciones adoptadas en el entorno punto a punto. El enfoque usual ha sido: (1) resolver el problema de la fiabilidad multipunto con protocolos extremo a extremo (protocolos de transporte); y (2) diseñar esquemas de recuperación de errores, *centralizados* (las retransmisiones se hacen desde un único punto, normalmente la fuente), y *globales* (los paquetes solicitados se vuelven a enviar al grupo completo); siendo todas éstas, soluciones típicas del

ámbito punto a punto. Este planteamiento ha conducido a protocolos poco escalables, con un alto retardo medio de distribución y que no optimizan el consumo de recursos de red.

Las deficiencias planteadas son muy relevantes. En primer lugar, es fundamental para un gran número de aplicaciones que un servicio multipunto para interredes escale de forma adecuada, y principalmente, con respecto al número de miembros y al tamaño de la interred. Segundo, si no se tiene especial cuidado a la hora de plantear un servicio fiable, el retardo medio de distribución puede verse incrementado de forma desorbitada al aumentar el número de miembros del grupo o con el tamaño de la interred. Y por último, la no optimización en el consumo de recursos de red es una deficiencia que adquiere una especial relevancia en el ámbito multipunto [TCD95], y sin embargo, hasta recientemente muy pocos autores la han tenido en consideración al diseñar servicios multipunto fiables; debe considerarse que los recursos de red precisados por una única comunicación multipunto pueden ser muy elevados si se comparan con los utilizados por una comunicación punto a punto, de aquí sea de vital importancia que los protocolos multipunto, y especialmente los diseñados para *redes de área extensa* (WAN[1]), traten de optimizar los recursos utilizados.

Definidas las principales deficiencias de las soluciones tradicionales al problema de la fiabilidad multipunto, se plantean los principales objetivos a perseguir:

- Proponer una solución lo más *genérica* posible, que cubra las necesidades de un amplio espectro de aplicaciones multipunto que precisan fiabilidad.

- Ofrecer buenas características de *escalabilidad* en dos dimensiones: número de miembros y tamaño de la interred.

- El servicio propuesto además de fiable debe ser *robusto*. Un protocolo que declara un fallo y aborta una comunicación, cuando se producen "ciertos" errores puede ser fiable pero no útil [BOG+94]. Por ejemplo, en una aplicación de distribución de software una propiedad de robustez deseable puede ser, la capacidad de continuar operando a pesar de que el grupo se divida debido a una partición temporal en la interred. Planteados estos casos, finalmente deberían ser las aplicaciones las que decidieran cuando la situación se ha deteriorado hasta tal punto que no tiene sentido continuar la comunicación.

[1] *Wide Area Network.*

- *Minimizar el retardo medio de distribución.* Algunas aplicaciones precisan al mismo tiempo un servicio multipunto, fiable y que exhiba un buen comportamiento en cuanto al retardo medio de distribución, dado que un incremento significativo de éste puede afectar al rendimiento de la aplicación, por ejemplo sistemas de Simulación Interactiva Distribuida [HSC95].

- *Optimizar el uso de los recursos de red,* aspecto que como ya se ha señalado es de gran relevancia en el entorno multipunto.

Las ideas básicas en las que se fundamenta la propuesta planteada en este trabajo son las siguientes:

- *Separar el servicio de fiabilidad de los servicios de ordenación y atomicidad.* Tras estudiar las principales calidades de servicio requeridas por las aplicaciones multipunto se ha observado que la fiabilidad puede considerarse ortogonal a la ordenación y a la atomicidad. Primeramente, existe una gran dispersión en las necesidades de orden, por otro lado las aplicaciones que precisan de fiabilidad no requieren siempre de las calidades de ordenación y atomicidad, y por último los mecanismos empleados para proporcionar estas tres calidades de servicio son diferentes. Consecuentemente no se obtienen ventajas significativas por el hecho de tratarlos conjuntamente, diseñando un protocolo que ofrezca fiabilidad y distintos grados de ordenación y atomicidad sino que por el contrario, puede dar como resultado un sistema monolítico poco flexible y con dificultades de evolución.

  Obviamente, si la aplicación precisa otras garantías de servicio además de fiabilidad, estás pueden ser proporcionadas o por un protocolo de nivel superior o por la propia aplicación. Como ya ha sido argumentado en [FJM+95] el proporcionar distintos grados de ordenación sobre un servicio fiable siempre es posible, y lo mismo ocurre con la atomicidad.

- Proporcionar *fiabilidad a nivel de red* en lugar de hacerlo mediante un protocolo extremo a extremo (protocolo de transporte). Esto va a permitir optimizar el uso de recursos de red y lograr una solución escalable respecto al número de miembros y al tamaño de la interred.

- El hecho de proporcionar fiabilidad con un protocolo de nivel de red no debe implicar que éste sea dependiente del algoritmo de encaminamiento multipunto utilizado. El lograr esta *independencia del algoritmo de encaminamiento multipunto subyacente* favorece a la generalidad del protocolo.

- ❒ Realizar la *retransmisión de los paquetes perdidos desde ciertos sistemas intermedios de la interred*, hecho que además de optimizar el uso de los recursos de red, hace posible una disminución del retardo medio de distribución.

- ❒ *Seguir el modelo de servicio multipunto propuesto por Deering* [Dee91] incorporando la garantía de fiabilidad. De aquí en adelante, cuando se hable de un servicio multipunto fiable que sigue el modelo de Deering (como se vio en la sección 2.3.1 este modelo proporciona una fiabilidad de "mejor esfuerzo"), se estará hablando de un servicio multipunto que sigue todas las proposiciones del modelo de Deering excepto la relativa a la fiabilidad.

  Como será analizado en el siguiente capítulo, el proporcionar un servicio fiable que sigue el modelo de Deering, y en concreto, aceptar el hecho de que la fuente no conozca a priori la identidad de los miembros del grupo, impone ciertas limitaciones al grado de fiabilidad ofrecido. La justificación a aceptar este modelo de servicio es su gran flexibilidad y generalidad, características que hacen que sea aplicable para muchas aplicaciones; adicionalmente este modelo permite que, siempre que sea necesario imponer restricciones como ejemplo que el grupo sea cerrado o estático, esto pueda hacerse mediante un protocolo de gestión de grupos de nivel superior [Dee91].

- ❒ El aspecto de *gestión de grupos* es ortogonal al problema de la fiabilidad, y además, está muy ligado a la semántica de la aplicación, de aquí que no haya sido tratado en este trabajo.

Capítulo 4.

NIVELES DE FIABILIDAD

En este capítulo se estudia el problema de la fiabilidad en el entorno multipunto. En primer lugar se definen distintos niveles de ésta de acuerdo con la variación de diferentes parámetros. Adicionalmente, se plantean las limitaciones a la fiabilidad derivadas de: (1) utilizar un protocolo orientado al receptor puro y (2) seguir el modelo de Deering.

4.1 Niveles de fiabilidad en el entorno multipunto

No todos los protocolos multipunto que se denominan de forma genérica *fiables*, proporcionan el mismo nivel de fiabilidad. Al estudiar la semántica de la fiabilidad en el entorno multipunto se plantean varios aspectos inexistentes en el ámbito punto a punto: (1) un grupo estará formado por n miembros (n destinos), de aquí que además de la semántica, los datos han sido recibidos por todos los miembros, sea posible definir ciertos grados de fiabilidad menos exigentes, por ejemplo, una fiabilidad grado k (siendo $k < n$) que garantiza, que de los n miembros del grupo, k han recibido todos los datos; (2) el número de receptores puede variar a lo largo de una conexión si el grupo es dinámico o, se mantiene fijo si el grupo es estático, estas dos posibilidades hacen que la semántica de la fiabilidad varíe dependiendo de si el grupo es estático o dinámico; (3) la actuación lógica ante, un fallo en un miembro o una partición en la interred, depende en cierta medida de la semántica de la aplicación. Por ejemplo, dada una aplicación de distribución de software que está enviando software a 500 puntos distintos, ¿es lógico que ante una partición en la interred que cause que dos puntos queden aislados, sea abortada la comunicación?.

Teniendo en cuenta los anteriores aspectos, se van a definir diversos grados de fiabilidad que se considera pueden ser apropiados para distintas aplicaciones multipunto:

- ***Fiabilidad en grupo estático.*** Un servicio que proporciona este nivel de fiabilidad garantiza, que cuando la comunicación finaliza con una liberación ordenada de la conexión, todos los paquetes enviados al grupo han sido recibidos en secuencia, sin pérdidas ni duplicados, por todos los miembros del grupo.

  Cuando se proporciona este servicio y, un miembro del grupo falla o hay una partición en la interred, la comunicación finaliza con un aborto de la conexión.

- ***Fiabilidad grado k en grupo estático.*** Un servicio que proporciona este nivel de fiabilidad garantiza, que cuando la comunicación finaliza con una liberación ordenada de la conexión, todos los paquetes enviados al grupo han sido recibidos en secuencia, sin pérdidas ni duplicados, por al menos *k* miembros del grupo.

  Si un miembro del grupo falla o hay una partición en la interred, y este hecho impide que al menos *k* miembros del grupo reciban todos los paquetes, la comunicación finaliza con un aborto de la conexión.

- ***Fiabilidad en grupo dinámico.*** Un servicio que proporciona este nivel de fiabilidad garantiza, que cuando la comunicación finaliza con una liberación ordenada de la conexión, cada miembro ha recibido todos los paquetes enviados al grupo mientras estaba suscrito a dicho grupo.

  Al producirse un fallo en un miembro o una partición, son posibles tres actuaciones: (1) continuar con la comunicación como si los miembros afectados se hubieran salido del grupo; (2) abortar la conexión, dado que lo normal cuando una aplicación se ajusta a la semántica de grupos dinámicos (por ejemplo servicios de distribución gratuitos), es que no sea importante quién pertenece al grupo en cada momento, y de aquí que esté permitido que los miembros entren o salgan cuando así lo deseen, no parece muy lógico abortar la conexión cuando se produce una partición o un fallo en un miembro; (3) continuar con la comunicación como si los miembros afectados se hubieran salido del grupo, pero adicionalmente indicar el hecho a la aplicación y dejar que ésta, de acuerdo con su propia semántica, decida si se debe abortar la conexión o continuar.

- ***Fiabilidad grado k en grupo dinámico.*** Un servicio que proporciona este nivel de fiabilidad garantiza, que cuando la comunicación finaliza con una liberación orde-

nada de la conexión, cada paquete enviado al grupo ha sido recibido en secuencia, sin pérdidas ni duplicados, por al menos *k* miembros. Aunque este nivel de fiabilidad no parece ser muy representativo podría resultar útil para algunas aplicaciones, como por ejemplo, una aplicación de enseñanza a distancia en la cual el profesor continúa impartiendo clase mientras haya al menos *k* alumnos atendiéndola.

Si un miembro del grupo falla o hay una partición en la interred, y este hecho impide que al menos *k* miembros continúen recibiendo los datos enviados al grupo, la comunicación finaliza con un aborto de la conexión.

En los niveles de fiabilidad anteriormente planteados todos los participantes de una comunicación (fuente o miembros), deben poder discernir si una comunicación ha finalizado con una liberación ordenada o con un aborto de la conexión.

Debe observarse que si un protocolo da un servicio de fiabilidad en grupo dinámico y la semántica de la aplicación determina que el grupo es estático, la aplicación puede emplear protocolos de nivel más alto para controlar, cuando los miembros pueden unirse o dejar un grupo dado [Dee91]. Consecuentemente, un servicio de fiabilidad en grupo dinámico es más genérico que un servicio de fiabilidad en grupo estático.

Sería posible definir otros niveles de fiabilidad, pero los aquí propuestos son lo suficientemente representativos, como para satisfacer las necesidades de fiabilidad de la mayoría de las aplicaciones multipunto que requieren este servicio.

La única forma de proporcionar los niveles de fiabilidad anteriormente expuestos, es haciendo que en todo momento la fuente conozca la identidad de los miembros (inicialmente antes de empezar a mandar los datos, y durante la transferencia si entra o sale algún miembro), y que ésta disponga de mecanismos que le permitan comprobar periódicamente el estado de cada uno de los miembros. De esta afirmación se extraen dos conclusiones importantes que son razonadas en las dos siguientes secciones:

1. Es imposible proporcionar los niveles de fiabilidad anteriormente expuestos con un protocolo orientado al receptor puro, dónde únicamente los receptores asumen la responsabilidad de detectar anomalías.

2. Es imposible proporcionar los niveles de fiabilidad anteriormente expuestos cuando se sigue el modelo de Deering, debido a que la fuente no conoce a priori la identidad de los miembros del grupo.

Debe observarse, que al igual que ocurre en el ámbito punto a punto, en el ámbito multipunto la fiabilidad ofrecida por un protocolo etiquetado como *fiable* siempre está sujeta a limitaciones diversas, tales como el código de detección de errores empleado; y por lo tanto no existe una fiabilidad total sino probabilidades más o menos elevadas de detectar los posibles fallos. Consecuentemente, el grado de fiabilidad es algo relativo y de aquí que se hable de protocolos más o menos fiables.

4.2 Fiabilidad versus protocolos orientados al receptor

Como se comentó en la sección 2.1.4, cuando se sigue este tipo de aproximación, únicamente los receptores (miembros) asumen la responsabilidad de detectar todas las posibles anomalías (pérdida de paquetes, fallo de la fuente, particiones en la interred, etc.). Siguiendo este planteamiento, entre otras, pueden darse las siguientes situaciones:

- Si un miembro pierde n paquetes consecutivos distribuidos al grupo, está situación se detectará cuando dicho miembro reciba el paquete $n+1$, pero sin embargo, la única forma de garantizar que en dicho momento todavía existan copias en algún sistema de los n paquetes perdidos (por ejemplo en la fuente), es guardando copia de *todos* los paquetes enviados al grupo mientras dura la comunicación. Obviamente, esta solución impone límites a la cantidad de información que se puede enviar y plantea problemas de optimización de recursos.

- Si un miembro pierde los últimos n paquetes distribuidos al grupo, no será capaz de detectarlo, dado que nunca recibirá un paquete fuera de secuencia que le avise de lo ocurrido. Obviamente, si no es consciente de la pérdida cuanto menos solicitará una retransmisión.

- Si un miembro pierde todos los paquetes distribuidos al grupo (por ejemplo por una partición), esta situación será imposible de detectar y por tanto de corregir.

- La fuente no puede distinguir entre, el fallo de un miembro o que todo ha funcionado correctamente, ya que en cualquiera de los dos casos dicho miembro no solicitará retransmisiones.

Las situaciones expuestas permiten concluir que, el seguir una aproximación orientada al receptor pura impone graves restricciones al grado de fiabilidad proporcionado.

4.3 Fiabilidad versus modelo de Deering

Uno de los principales aciertos del modelo de servicio planteado por Deering es el grado de indirección entre la fuente y los miembros, tal como ha sido valorado por distintos autores [FJM+95]. Esta indirección permite que la fuente pueda mandar datos a un grupo sin conocer a priori quienes son los miembros de dicho grupo (direccionamiento lógico), simplemente utilizando la dirección del grupo, y al mismo tiempo permite que los miembros se suscriban a un grupo sin tener que conocer a priori quienes serán las fuentes de dicho grupo. Este nivel de indirección es de suma importancia para muchas aplicaciones multipunto; por ejemplo, una aplicación de distribución de noticias deportivas empieza a distribuir noticias a un determinado grupo sin ser preciso un conocimiento previo sobre que sistemas desean recibir este tipo de noticias, y por lo tanto, se han suscrito previamente al grupo, y al mismo tiempo, los sistemas que desean recibir estas noticias se suscriben a un determinado grupo, pero no tienen por qué conocer quiénes serán las fuentes; como contrapartida, este modelo limita el nivel de fiabilidad que es posible proporcionar.

Para razonar la anterior afirmación se verá qué problemas se plantean en un protocolo que sigue una aproximación orientada al emisor, y permite el grado de indirección fuentes-miembros propuesto por el modelo de Deering. En este tipo de protocolos es responsabilidad del emisor (fuente) comprobar que todos los destinos reciben copia de los paquetes recibidos, así como de detectar las posibles anomalías tales como el fallo de un miembro o una partición en la interred. La fuente antes de descartar la copia que mantiene de cada paquete, espera a recibir confirmación de que cada uno de los miembros hayan recibido dicho paquete.

Dado que la fuente no conoce a priori quiénes son los miembros del grupo, tampoco sabe de quiénes debe esperar confirmación, sin embargo, es factible incluir un mecanismo que permita, que cuando cada miembro reciba un primer paquete de datos avise a la fuente de su pertenencia al grupo, y de que por lo tanto, debe ser tenido en cuenta antes de descartar la copia mantenida de los paquetes previamente enviados. Este enfoque es válido, pero plantea la posibilidad de que ocurra una implosión en la fuente, o bien inicialmente cuando cada miembro al recibir un primer paquete avisa a ésta de su existencia, o bien posteriormente al confirmar cada paquete recibido.

Un refinamiento a la solución anteriormente planteada es permitir que la fuente delegue parte de su responsabilidad en otros sistemas (sistemas con responsabilidad parcial), que a su vez pueden delegarla parcialmente en otros, de esta forma el conocimiento sobre los miembros y la responsabilidad sobre éstos, estará distribuida. Como resultado se obtie-

ne una estructura jerárquica en árbol (ver Figura 4.1) en la cual los miembros serán las hojas del árbol y la fuente la raíz. Cada sistema es responsable de que, todos los miembros por debajo de él en el árbol obtengan una copia de cada paquete, y por tanto la fuente (raíz del árbol) asume la responsabilidad total. El funcionamiento será el siguiente: (1) cada miembro al recibir el primer paquete avisa de su existencia al sistema que tiene responsabilidad directa sobre él; (2) un sistema con responsabilidad parcial envía un asentimiento positivo hacia la fuente únicamente cuando sabe que todos los miembros dependientes de él, ya tienen una copia del paquete que está siendo confirmado. Por extensión, cuando la fuente reciba un asentimiento desde cada una de las ramas, sabrá que todos los miembros del grupo han recibido una copia del paquete asentido.

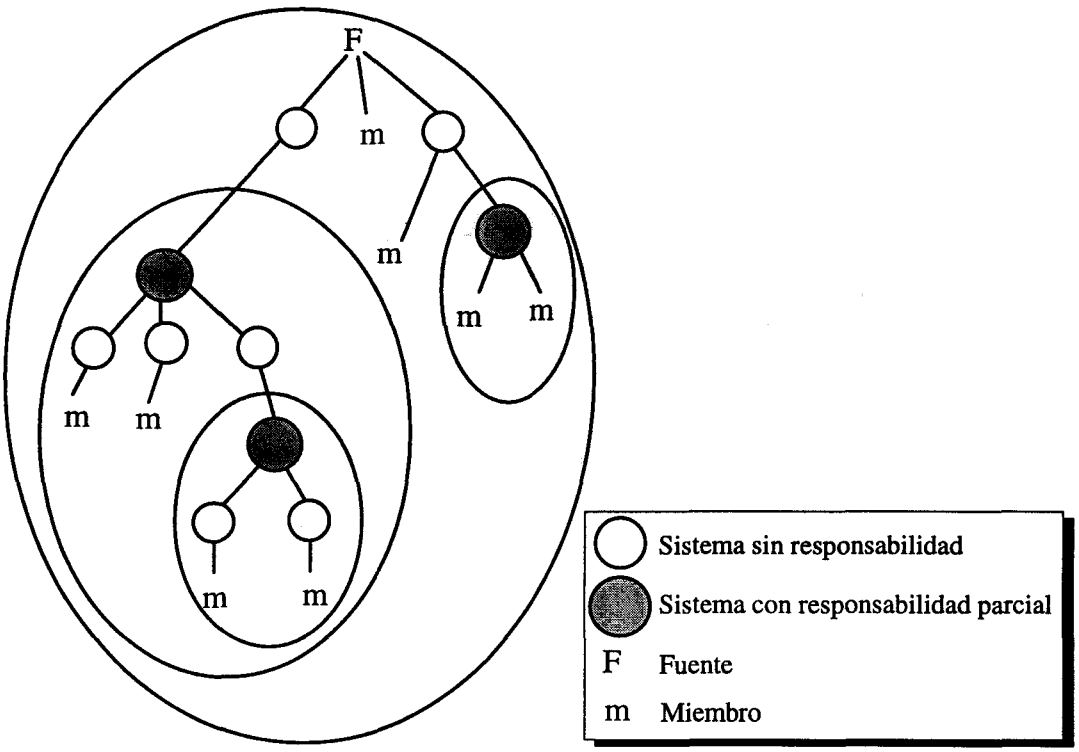


Figura 4.1: Árbol de delegación de responsabilidad

En la solución esbozada, hasta que cada sistema no conoce quiénes están bajo su responsabilidad, no puede garantizar que éstos han recibido los paquetes enviados al grupo, y por otro lado, hasta que un miembro no recibe un primer paquete no puede avisar a su responsable directo de su pertenencia al grupo; estos hechos posibilitan que un miembro que pertenecía al grupo desde antes de que la fuente empezara a distribuir datos, pierda los *n* primeros paquetes y cuando recibe el primer paquete, y consiguientemente avisa de su pertenencia al grupo, ya no exista ninguna copia en la red de los paquetes que se habían

perdido. Este tipo de situaciones pueden darse inicialmente, al empezar la fuente a mandar datos, y posteriormente si es de nuevo necesario que cada sistema aprenda quienes están bajo su responsabilidad (por ejemplo debido a la caída de un sistema que tenía responsabilidad parcial). Como consecuencia, el hecho de que la fuente no conozca a priori quiénes son los miembros del grupo limita el grado de fiabilidad proporcionado.

La solución al problema de la fiabilidad multipunto planteada en este trabajo, sigue el modelo de Deering y utiliza un protocolo orientado al emisor; en este caso, la fuente delega responsabilidades en ciertos sistemas intermedios (encaminadores) de la interred. Como ya se ha analizado, estas condiciones limitan el grado de fiabilidad proporcionado, sin embargo sí es posible:

- *Incluir mecanismos que hagan que la probabilidad de pérdidas sea muy pequeña.* Para reducir esta probabilidad, en los periodos donde es más probable que el conocimiento sobre los miembros sea incompleto, se adoptan las siguientes medidas correctivas:

  - Retransmitir más copias de los paquetes enviados en dichos periodos, con el fin de favorecer el hecho de que cada miembro reciba lo antes posible el primer paquete, y por lo tanto avise de su existencia.

  - Mantener por más tiempo copia de los paquetes enviados en dichos periodos. Esto tiene por finalidad aumentar la probabilidad de que cuando un miembro reciba el primer paquete, si había perdido ciertos paquetes anteriores y solicita su retransmisión, todavía exista copia de éstos en algún sistema.

  Como puede observarse, la reducción en la probabilidad de pérdidas y consecuentemente el incremento en fiabilidad, se hace a costa de utilizar más recursos de red en los periodos donde es más probable que el conocimiento sobre los miembros sea incompleto. Esto no constituye un gran problema pues estos periodos no serán frecuentes.

- *Indicar al usuario del servicio* la existencia de los periodos en los que es posible que el conocimiento sobre los miembros sea incompleto, y por tanto, el grado de fiabilidad proporcionado estará limitado. Este planteamiento de indicar determinados sucesos que pueden afectar a la fiabilidad, ya ha sido adoptado por otros protocolos de amplia difusión. Por ejemplo X.25 no garantiza que no haya pérdida de paquetes cuando se produce un reinicio pero lo señaliza al nivel superior.

El coste de estas indicaciones es muy bajo, mientras que al mismo tiempo, el nivel superior puede utilizar estas indicaciones para ejecutar determinados procedimientos que mejoren la fiabilidad o simplemente puede despreciarlas.

Visto lo anterior se plantean nuevos niveles de fiabilidad:

- ***Fiabilidad en grupo estático limitada.*** Un servicio que proporciona este nivel de fiabilidad asegura, con probabilidad elevada, que cuando la comunicación finaliza con una liberación ordenada de la conexión, todos los paquetes enviados al grupo han sido recibidos en secuencia, sin pérdidas ni duplicados, por todos los miembros del grupo, y que todas las situaciones que podían causar la pérdida de uno o más paquetes en algún miembro han sido indicadas al usuario del servicio.

  Cuando se proporciona este servicio y, un miembro del grupo falla o hay una partición en la interred, la comunicación finaliza con un aborto de la conexión.

- ***Fiabilidad grado k en grupo estático limitada.*** Un servicio que proporciona este nivel de fiabilidad asegura, con probabilidad elevada, que cuando la comunicación finaliza con una liberación ordenada de la conexión, todos los paquetes enviados al grupo han sido recibidos en secuencia, sin pérdidas ni duplicados, por al menos k miembros del grupo, y que todas las situaciones que podían causar la pérdida de uno o más paquetes en algún miembro han sido indicadas al usuario del servicio.

  Si un miembro del grupo falla o hay una partición en la interred, y este hecho impide que al menos k miembros del grupo reciban todos los paquetes, la comunicación finaliza con un aborto de la conexión.

- ***Fiabilidad en grupo dinámico limitada.*** Un servicio que proporciona este nivel de fiabilidad asegura, con probabilidad elevada, que cuando la comunicación finaliza con una liberación ordenada de la conexión, para cada miembro del grupo se cumple que ha recibido todos los paquetes enviados al grupo mientras estaba suscrito a dicho grupo, y que todas las situaciones que podían causar la pérdida de uno o más paquetes en algún miembro han sido indicadas al usuario del servicio.

  Al detectar el fallo de un miembro o una partición en la interred son posibles tres actuaciones: (1) abortar la conexión; esta opción no es muy lógica, dado que cuando una aplicación se ajusta a la semántica de grupos dinámicos (por ejemplo servicios de distribución gratuitos) carece de importancia para la fuente quién pertenece al grupo en cada momento y de aquí que esté permitido que los miembros entren o

salgan en cualquier momento; (2) continuar con la comunicación como si los miembros afectados se hubieran salido del grupo; (3) continuar con la comunicación como si los miembros afectados se hubieran salido del grupo pero adicionalmente indicar a la aplicación el hecho, y dejar que ésta, de acuerdo con su propia semántica, decida si se debe abortar la conexión o continuar.

- ***Fiabilidad grado k en grupo dinámico limitada.*** Un servicio que proporciona este nivel de fiabilidad asegura, con probabilidad elevada, que cuando la comunicación finaliza con una liberación ordenada de la conexión, cada paquete enviado al grupo ha sido recibido en secuencia, sin pérdidas ni duplicados, por al menos k miembros y que todas las situaciones que podían causar la pérdida de uno o más paquetes en algún miembro han sido indicadas al usuario del servicio. Aunque este nivel de fiabilidad no parece ser muy representativo podría resultar útil para algunas aplicaciones, como por ejemplo, una aplicación de enseñanza a distancia en la cual el profesor continúa impartiendo clase mientras haya al menos k alumnos atendiéndola.

  Si un miembro del grupo falla o hay una partición en la interred, y este hecho impide que al menos k miembros continúen recibiendo los datos enviados al grupo, la comunicación finaliza con un aborto de la conexión.

Debe observarse que no se considera una pérdida de fiabilidad el hecho de que si el grupo es dinámico (por la propia semántica de la aplicación), y permite que un miembro se suscriba al grupo en el momento que así lo desee, en lugar de recibir a partir del paquete n empiece a recibir a partir del paquete $n+k$.

La solución propuesta en este trabajo proporciona un nivel de *fiabilidad en grupo dinámico limitada,* y el comportamiento cuando se detecta el fallo de un miembro o una partición, es continuar con la comunicación como si los miembros afectados se hubieran salido del grupo, y adicionalmente, indicar a la aplicación el hecho y dejar que ésta, de acuerdo con su propia semántica, decida si se debe abortar la conexión o continuar.

Aunque la definición actual del protocolo multipunto fiable aquí propuesto proporcione un servicio de *fiabilidad en grupo dinámico limitada,* es relativamente sencillo incorporar mecanismos que sin seguir el modelo de Deering, obtengan *fiabilidad en grupo dinámico*; para validar esta afirmación en la sección 5.11, se esbozan dichos mecanismos.

Capítulo 5.

DISEÑO DEL PROTOCOLO RMNP

Este capítulo describe el proceso de diseño del protocolo RMNP *(Reliable Multicast Network Protocol)*, el cual constituye la principal contribución original de este trabajo. A lo largo del capítulo se van exponiendo diversas alternativas para resolver los distintos aspectos del protocolo, analizando la idoneidad de éstas para lograr los objetivos planteados en el capítulo 3. Posteriormente, en el apéndice A se presenta las unidades de datos del protocolo, y en el apéndice B se detalla de forma más precisa la operación del protocolo mediante pseudocódigo.

RMNP es un *protocolo multipunto fiable* que opera a nivel de red. Esto significa que, a pesar de errores de transmisión, saturación de buffers, fallos en enlaces y encaminadores, y en general, cambios en la topología de la interred que causan una modificación en el árbol de distribución multipunto sin particionar la interred, todos los paquetes enviados a un grupo serán recibidos por los miembros de dicho grupo en secuencia, sin pérdidas ni duplicación.

La definición actual del RMNP sigue el modelo de grupo de Deering, de aquí que en determinadas condiciones sea imposible garantizar que no se ha producido una pérdida de paquetes (ver sección 4.3), sin embargo, estas situaciones serán señalizadas por RMNP al usuario del servicio. De este modo, en el caso de que el nivel de fiabilidad proporcionado por RMNP sea inaceptable para la aplicación, un nivel superior a RMNP (sin seguir dicho modelo de grupo), puede incorporar los procedimientos necesarios para resolver las posibles pérdidas de paquetes.

El modelo de red considerado al diseñar el RMNP es el de una interred que opera en modo datagrama, compuesta de una colección de RALs interconectadas por encaminadores, enlaces y subredes de cualquier tipo.

5.1 Interacción con otros protocolos

RMNP *interactúa* a nivel de red con un servicio de distribución de datagramas multipunto, que proporciona un servicio sin garantía de calidad, y con una arquitectura de encaminamiento multipunto (ver Figura 5.1). En concreto, RMNP introduce en el nivel de red una serie de procedimientos adicionales que mejoran el servicio ofrecido por los anteriores módulos, incorporando fiabilidad. Cuando se afirma que RMNP *interactúa* con dichos módulos, se desea reflejar que por un lado RMNP utiliza los servicios de éstos, pero por otro lado también altera su comportamiento, por ejemplo modificando el reenvío. Se asume que el mencionado servicio de distribución de datagramas multipunto, también será capaz de enviar un datagrama punto a punto cuando la dirección destino sea una dirección individual en lugar de una dirección de grupo.

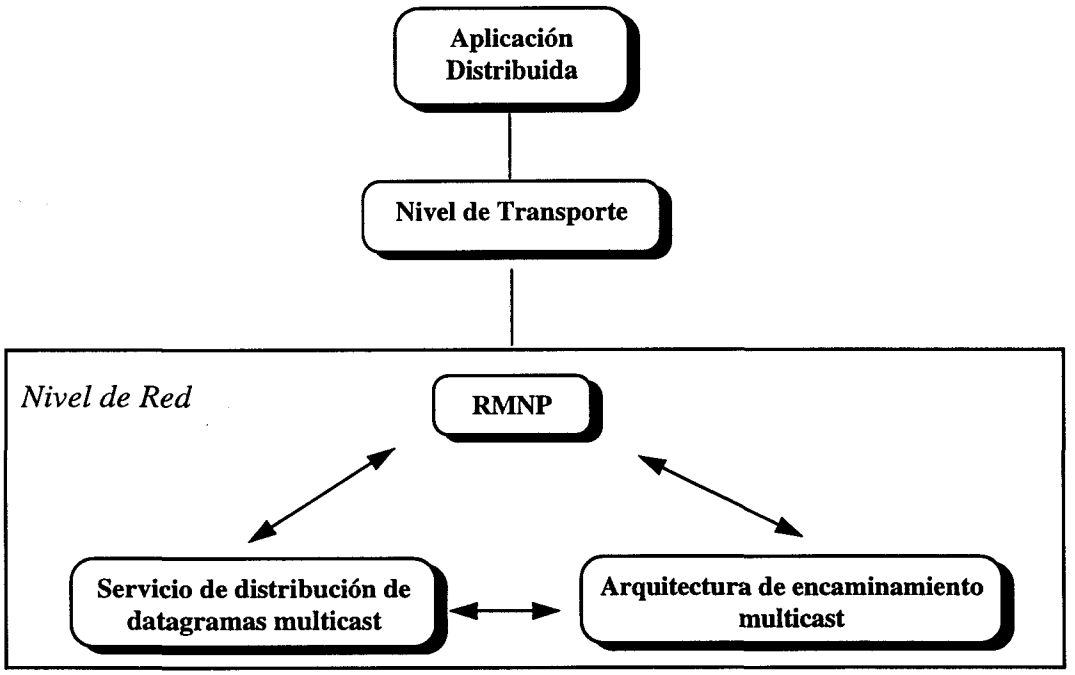


Figura 5.1: Interacción con otros protocolos

Como puede deducirse no se está siguiendo el modelo de referencia de ISO [ISO84], sin embargo, este comportamiento no constituye una excepción, sino que existen muchos

ejemplos de arquitecturas de comunicaciones ampliamente usadas que siguen la misma aproximación (por ejemplo, ICMP[1] usa los servicios de IP y al mismo tiempo, IP también utiliza los servicios de ICMP).

RMNP opera con arquitecturas de encaminamiento multipunto basadas en árboles de distribución, sin embargo no está restringido a un único tipo de arquitectura. En concreto, la arquitectura de encaminamiento multipunto puede estar basada en árboles fuente, en árboles compartidos o en una combinación de ambos.

5.2 Terminología y consideraciones previas

En esta sección se definen los términos usados, y se hacen algunas consideraciones previas, con el fin de clarificar la exposición del RMNP:

- **Sistema RMNP.** Término genérico que engloba a cualquier dispositivo que dispone de capacidades RMNP. Dicho dispositivo puede ser un sistema final o un sistema intermedio. Los sistemas RMNP pueden clasificarse de forma más precisa en:

  - **Fuente.** Sistema final que *envía* paquetes multipunto a un grupo.

  - **Miembro.** Sistema final que pertenece a un grupo, y como tal *recibe* los paquetes enviados a dicho grupo. Como ya se ha comentado en la sección 2.3.1 una fuente no tiene por que ser miembro del grupo.

  - **Encaminador básico (BR).** Encaminador que incorpora todas las funcionalidades del RMNP excepto la del *almacenamiento intermedio* y la *retransmisión* (ver secciones 5.4.3 y 5.6.3).

  - **Encaminador Almacén (SR).** Encaminador con funcionalidad RMNP completa, incluyendo el *almacenamiento intermedio* y la *retransmisión* (ver secciones 5.4.3 y 5.6.3). Debe notarse que aunque un BR y un SR incorporan el mismo conjunto de funciones básicas en algunos casos para la misma función pueden variar los procedimientos.

[1] *Internet Control Message Protocol.*

- **Encaminador no-RMNP.** Encaminador que no dispone de ningún tipo de funcionalidad RMNP.

- **Encaminador RMNP.** Encaminador que dispone de funcionalidad RMNP. Con este término se engloban tanto los SRs como los BRs.

- **Árbol de distribución.** Árbol utilizado para distribuir a todos los miembros de un grupo, los datos enviados por una fuente. El árbol de distribución *viene determinado* por la arquitectura de encaminamiento multipunto subyacente. Como consecuencia, la información de estado intercambiada y almacenada en cada nodo, para crear y mantener dicho árbol, será independiente del nivel RMNP.

  En el caso de que la arquitectura de encaminamiento multipunto utilice árboles fuente (ver sección 2.3.2.1) se cumplirá lo siguiente:

  1. Existe un árbol de distribución por fuente y grupo.

  2. La raíz de cada árbol de distribución está en la fuente.

  3. Cuando la fuente envía datos al grupo, la distribución por el árbol empieza a partir de la raíz.

  En el caso de que la arquitectura de encaminamiento multipunto utilice árboles compartidos (ver sección 2.3.2.2) se cumplirá lo siguiente:

  1. Existe un árbol de distribución por grupo que será compartido por todas las fuentes.

  2. La raíz del árbol de distribución de un grupo está en un determinado encaminador, elegido para realizar tal función (por ejemplo el encaminador "core" en el caso de utilizarse CBT).

  3. La fuente puede no pertenecer al árbol de distribución. Esto ocurrirá si la fuente no es miembro del grupo y no está en el camino hacia ningún miembro.

  4. Cuando una fuente envía datos al grupo, la distribución por el árbol empieza a partir de un determinado nodo, que no tiene porqué coincidir con la raíz del árbol.

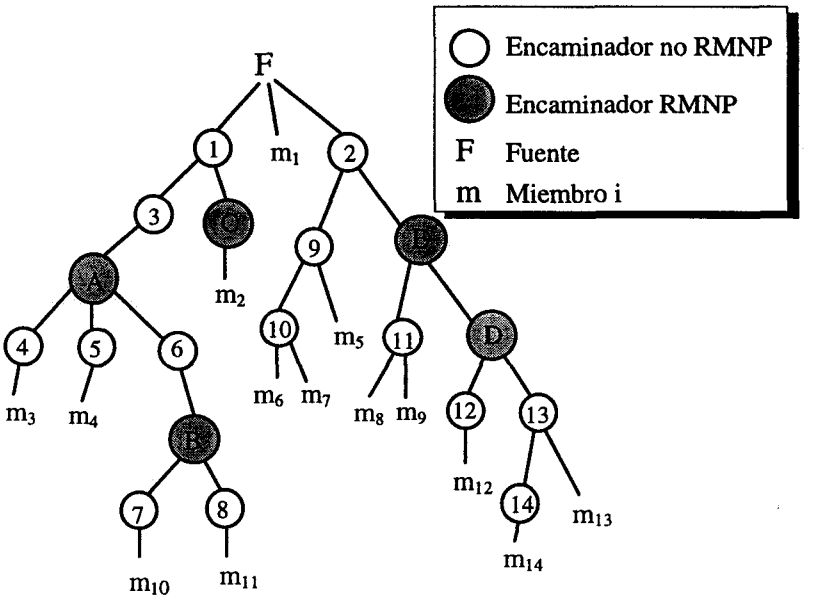


Figura 5.2: Árbol de distribución

En la Figura 5.2 se presenta un ejemplo de árbol de distribución, éste muestra las relaciones *padre-hijo*. La figura no refleja el número de interfaces que tiene un nodo, de forma que varios hijos pueden compartir el mismo interfaz físico (por ejemplo, si un nodo esta conectado a una subred).

Cada *enlace* entre nodos es una de las siguientes posibilidades:

- Una línea punto a punto.
- Una subred. Opcionalmente, ésta puede proporcionar un servicio multipunto[2].
- Un túnel[3].

[2] Si la subred ofrece un servicio multipunto, éste es normalmente utilizado en la distribución de datos por el árbol de distribución. Este hecho es transparente al nivel RMNP.

[3] Procedimiento usado para "puentear" partes de una interred que no ofrecen un servicio multipunto o siguen políticas distintas. El mecanismo normalmente usado en los túneles, es el de la encapsulación. Esta consiste en envolver la cabecera original del datagrama con una nueva cabecera. En la cabecera exterior las direcciones origen y destino identifican a los puntos extremos del túnel. En la cabecera interior las direcciones origen y destino identifican al origen y el destino reales del datagrama.

Cada *nodo* es una de las siguientes posibilidades:

- Una fuente.
- Un miembro[4].
- Un encaminador RMNP.
- Un encaminador no-RMNP.

Como es lógico, todo encaminador perteneciente al árbol de distribución, sea RMNP o no, tendrá implementado el protocolo de encaminamiento multipunto.

- **Ascendientes en el árbol de distribución**. Dado un nodo *i* perteneciente al árbol de distribución sus ascendientes serán todos aquellos nodos que están en el camino entre la fuente y dicho nodo *i* (por ejemplo, en el caso mostrado en la Figura 5.2 serán ascendientes del nodo 4, los nodos: A,3,1,F).

- **Descendientes en el árbol de distribución**. Dado un nodo *i* perteneciente al árbol de distribución será su descendiente, todo aquel nodo *k* para el que se cumple: *i* pertenece al camino que une la fuente y dicho nodo k (por ejemplo, en el caso mostrado en la Figura 5.2 serán descendientes del nodo D, los nodos: 12, 13, 14, m_{12}, m_{13}, m_{14}).

- **Padre en el árbol de distribución**. El padre de un nodo *i* será aquel ascendiente que está situado a un sólo salto.

- **Hijos en el árbol de distribución**. Los hijos de un nodo *i* serán aquellos descendientes que están situados a un sólo salto.

- **Padre RMNP**. Para un determinado par (grupo, fuente) el nodo *i* será padre RMNP del nodo *k*, si se cumple que los paquetes enviados por la fuente *f* al grupo *g* cuando llegan al nodo *k* el último sistema RMNP que han visitado es el nodo *i*.

[4] Algunos algoritmos de encaminamiento multipunto suponen que los miembros siempre están conectados a subredes que ofrecen un servicio multipunto, y como consecuencia, los encaminadores que tienen miembros a un sólo salto utilizan dicho servicio para distribuirles datos, de este modo dichos encaminadores lo único que necesitan conocer es si tienen o no miembros a un sólo salto, mientras que la identidad de éstos es irrelevante. Como se verá a lo largo del capítulo este hecho es transparente al nivel RMNP.

❒ **Hijo RMNP.** Para un determinado par (grupo, fuente) el nodo *k* será hijo RMNP del nodo *i*, si se cumple que los paquetes multipunto enviados por la fuente *f* al grupo *g*, al llegar al nodo *k*, el último sistema RMNP que han visitado es el nodo *i*

❒ **Árbol RMNP.** Árbol definido por las relaciones *padre-hijo* RMNP y utilizado para procesar las contestaciones enviadas por los miembros hacia la fuente. El árbol RMNP está formado únicamente por sistemas RMNP. De las definiciones de padre e hijo RMNP se puede deducir que existe un árbol RMNP por cada par (grupo, fuente), y la raíz de cada uno de estos árboles será la fuente correspondiente.

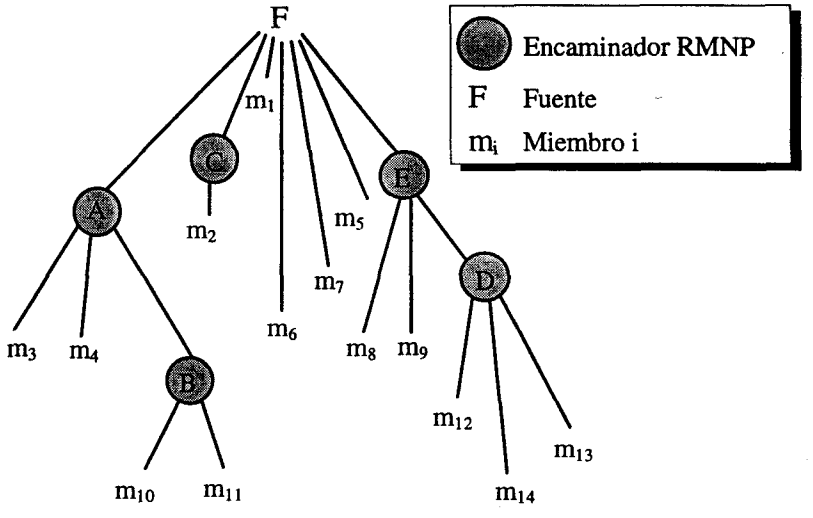


Figura 5.3: Árbol RMNP

En la Figura 5.3 se muestra un ejemplo de árbol RMNP, éste muestra las relaciones *padre-hijo* RMNP. La figura no muestra el número de interfaces que tiene un nodo, de aquí que varios hijos puedan compartir el mismo interfaz físico de salida (por ejemplo, si un nodo esta conectado a una subred). Éste sería el árbol RMNP correspondiente al árbol de distribución mostrado en la Figura 5.2. En el caso de que todos los encaminadores pertenecientes al árbol de distribución tuvieran funcionalidad RMNP, éste y el árbol RMNP coincidirían.

En un árbol RMNP, los nodos son:

- Sistemas RMNP.

Cada enlace entre nodos es una de las siguientes posibilidades:

- Una línea punto a punto.
- Una subred. Opcionalmente, ésta puede proporcionar un servicio multipunto.
- Un túnel.
- Una combinación de enlaces y nodos que no contiene ningún sistema RMNP.

Como se analizará más detalladamente en la sección 5.7.1, un árbol RMNP se crea inicialmente en el establecimiento de la conexión, y posteriormente irá actualizándose con la entrada/salida de miembros en la fase de transferencia de datos (ver sección 5.7.3).

Identificador del grupo	Dirección de la fuente	Dirección del padre	Interfaz hacia el padre	Interfaz en el padre (*)	Dirección del hijo 1	Interfaz hacia el hijo 1
					...	...
					Dirección del hijo n	Interfaz hacia el hijo n

(*) Cada sistema RMNP conoce el identificador del interfaz que usa su padre cuando le envía paquetes.

Figura 5.4: Entrada en la TIB

La información de estado necesaria para mantener los árboles RMNP se almacena en la **Base de información de árboles RMNP** (TIB). Un sistema RMNP puede pertenecer a varios árboles RMNP, y consecuentemente, dispondrá de una entrada en la TIB para cada uno de dichos árboles. En la Figura 5.4 se muestra el formato de una entrada en la TIB. Como se verá posteriormente, el hecho de que cada sistema guarde el identificador del interfaz que usa su padre para mandarle paquetes, tiene como finalidad hacer posible la detección de cambios en el árbol de distribución.

Todos los sistemas RMNP disponen de una TIB, con las particularidades de que las entradas en la TIB de la fuente tendrán vacía la información relativa al padre, y las entradas en la TIB de los miembros tendrán vacía la información relativa a los hijos.

- **Ascendientes RMNP**. Dado un nodo *i* perteneciente al árbol RMNP, sus ascendientes serán todos aquellos nodos RMNP que están en el camino entre la fuente y dicho nodo *i* (por ejemplo, en el caso mostrado en la Figura 5.3 serán ascendientes del nodo B, los nodos: A, F).

- **Descendientes RMNP**. Dado un nodo *i* perteneciente al árbol RMNP será su descendiente, todo aquel nodo RMNP *k* para el que se cumpla: *i* pertenece al camino que une la fuente y dicho nodo k (por ejemplo, en el caso mostrado en la Figura 5.3 serán descendientes del nodo A, los nodos: B, m_3, m_4, m_{10}, m_{11}).

- **Sistema Retransmisor**. Sistema RMNP que dispone de las funciones de almacenamiento y retransmisión. En concreto, son sistemas retransmisores la fuente y los SRs.

- **Sistema Padre**. Sistema RMNP que tiene hijos RMNP. En concreto son sistemas padre, la fuente y los encaminadores RMNP.

- **Sistema Hijo**. Sistema RMNP que tiene padre. En concreto son sistemas hijo, los encaminadores y los miembros RMNP.

- **Interfaz Padre**. Interfaz por el que está accesible el padre.

- **Interfaz Hijo**. Interfaz por el que están accesibles uno o más hijos RMNP.

Para comprender el funcionamiento del protocolo es preciso tener en consideración que los paquetes enviados por la fuente a los miembros del grupo se propagan a través del árbol de distribución utilizando los servicios del nivel de encaminamiento multipunto. Sin embargo, los paquetes enviados por los miembros hacia la fuente siguen el árbol RMNP, de forma que cuando un nodo recibe un paquete RMNP con destino a la fuente se lo envía a su padre RMNP como un paquete punto a punto.

El comportamiento de RMNP con respecto a cada par (grupo, fuente) es totalmente independiente. Para cada uno de estos pares existirá un árbol RMNP y un árbol de distribución asociados. Como consecuencia, la descripción del protocolo se hará para un único par (grupo, fuente) y sus árboles asociados.

5.3 Introducción al RMNP

5.3.1 Mecanismos utilizados para proporcionar fiabilidad

Los principales mecanismos empleados para proporcionar un servicio multipunto fiable son los siguientes:

- **Asentimientos positivos con retransmisiones.** Esta técnica obliga a que los miembros envíen a la fuente asentimientos positivos, confirmando la recepción correcta de los datos. La fuente guarda una copia de cada paquete enviado, que no descartará hasta haber recibido de los miembros los correspondientes asentimientos positivos. Otra operación realizada por la fuente antes de mandar un paquete multipunto, es poner en marcha un temporizador. En caso de que éste expire antes de haber recibido los correspondientes asentimientos positivos, la fuente retransmitirá el paquete multipunto.

  A diferencia de otras propuestas que utilizan asentimientos positivos con retransmisiones, los asentimientos positivos provenientes de los *n* miembros del grupo (*n* puede ser un valor muy grande incluso del orden de millares), no se mandan como mensajes punto a punto a la fuente [CP88], ni tampoco como un mensaje multipunto a todos los miembros del grupo [Whe95]. En RMNP, los asentimientos positivos se van propagando por el árbol RMNP hacia la fuente y cada encaminador que pertenece a dicho árbol irá realizando sobre los asentimientos positivos recibidos, un **proceso de agregación.** De este modo, si un encaminador tiene *m* hijos RMNP y ha recibido los *m* asentimientos positivos asociados a un mismo paquete enviará a su padre RMNP un único asentimiento. Como consecuencia de todo esto, la fuente recibe un único asentimiento por cada uno de sus hijos RMNP. Este planteamiento tiene ventajas importantes:

  - Evita un problema de *implosión* de asentimientos positivos en la fuente, característico de los esquemas que utilizan asentimientos positivos con retransmisiones.

  - Realiza una *reducción del trafico*, con el correspondiente ahorro en proceso y ancho de banda. Como ya se ha comentado, los encaminadores envían a su padre RMNP un único asentimiento en lugar de los *m* asentimientos positivos recibidos, uno por cada uno de sus hijos RMNP. De este modo, este proceso

de agregación logra una reducción exponencial de orden *m* de los asentimientos positivos enviados hacia la fuente (siendo *m* el grado medio del árbol RMNP).

El realizar un proceso de agregación sobre los asentimientos positivos tiene dos inconvenientes: en primer lugar, se debe obligar a todos los asentimientos positivos a seguir el mismo árbol RMNP y, como consecuencia, cuando haya un cambio en el árbol RMNP mientras esté en curso la confirmación de una serie de paquetes, debe ponerse en marcha una fase de reinicio. En segundo lugar, la agregación tiene un gasto en proceso y en memoria (se necesita más información de estado) en los encaminadores.

- **Solicitudes explícitas de retransmisión**. Uno de los principales objetivos en el diseño del RMNP es, que la retransmisión de los paquetes perdidos se haga a la mayor brevedad posible, para de este modo disminuir el retardo medio de distribución.

  Este planteamiento lleva a introducir mensajes que permitan solicitar explícitamente la retransmisión de los paquetes perdidos (asentimientos negativos). Otra opción de diseño considerada fue, que las retransmisiones de paquetes se hicieran únicamente cuando se vencieran los temporizadores asociados, como ocurre en otros protocolos fiables muy extendidos como el TCP. Esto simplificaría el protocolo pero, tiene un inconveniente fundamental: en el caso de no disponer de un mecanismo para solicitar explícitamente retransmisiones, éstas empezarían muy tarde, lo cual aumentaría el retardo medio de distribución. Esto viene causado por dos factores, por un lado, se deben evitar retransmisiones innecesarias que provocan un desperdicio de ancho de banda y de capacidad de proceso; por otro lado, en el escenario planteado algunos miembros del grupo pueden estar muy cercanos pero otros pueden estar a muchos saltos, por lo tanto, los temporizadores asociados a las retransmisiones deben tener valores bastante altos.

  Una segunda consecuencia del objetivo antes planteado es permitir que además de los miembros, los encaminadores RMNP puedan solicitar retransmisiones. Los encaminadores RMNP llevan control de los números de secuencia de los paquetes reenviados a sus hijos y, cuando detectan una pérdida, detienen el reenvío y solicitan la retransmisión a partir del paquete perdido. Este diseño permite detectar antes la pérdida de paquetes, adelantando la solicitud de retransmisión y por lo tanto el proceso de retransmisión.

Los asentimientos negativos se van propagando por el árbol RMNP hacia la fuente, y cada BR que pertenece a dicho árbol, irá realizando sobre los asentimientos negativos recibidos un **proceso de filtrado.** De este modo, si un BR ha solicitado y tiene pendiente de satisfacer la retransmisión desde el paquete *n*, y recibe desde un hijo RMNP un asentimiento negativo, únicamente lo reenvía hacia su propio padre RMNP si dicho asentimiento es más restrictivo (solicita la retransmisión de un paquete anterior al *n*); en caso contrario será descartado. Las principales ventajas de este proceso de filtrado son:

- Evita un problema de *implosión* de asentimientos negativos en los puntos de retransmisión (la fuente y los SRs).

- Produce una *reducción de trafico*, con el correspondiente ahorro en proceso y ancho de banda.

❒ **Retransmisión desde puntos intermedios.** El RMNP incluye un planteamiento novedoso en la forma de tratar la retransmisión de los paquetes perdidos. Las retransmisiones no se realizan desde un único punto (por ejemplo la fuente), sino que determinados encaminadores intermedios, los SRs, también disponen de la funcionalidad necesaria para realizar retransmisiones. Para hacer esto posible, la fuente y los SRs guardan copia de los paquetes pendientes de confirmación. Las ventajas de este planteamiento son las siguientes:

- El retardo causado por la retransmisión es menor, pues ésta se hará desde el SR más cercano al punto donde se originó el problema (se perdió el paquete). Como consecuencia de esto, hay un decremento en el retardo medio de distribución. Esta ventaja se acentúa en interredes muy extensas.

- Se ahorra ancho de banda y proceso porque los paquetes retransmitidos no necesitan ser distribuidos por todo el árbol de distribución, sino solamente por el subárbol con raíz en el punto de retransmisión más cercano.

- Distintos SRs pueden estar haciendo retransmisiones de diferentes paquetes al mismo tiempo, lográndose un cierto grado de paralelismo en el proceso de retransmisión.

El principal inconveniente de esta propuesta es el gasto en memoria. Los *n* paquetes pendientes de confirmar estarán almacenados en la fuente y en todos los SRs pertenecientes al árbol RMNP. Sin embargo, debe tenerse en cuenta que todas las

soluciones que reducen el retardo medio de distribución imponen un gasto adicional en memoria. Esto es debido a que la forma de disminuir significativamente el retardo medio de distribución es utilizando un esquema de retransmisiones distribuido, en el que se guarden copias de los paquetes pendientes de confirmación en *n* puntos, y haciendo las retransmisiones desde un punto cercano al lugar donde se originó el problema [YGS95,LS96].

5.3.2 Control de flujo

RMNP incorpora un esquema multiventana como mecanismo de control de flujo. Cada sistema retransmisor (fuente o SR), mantiene una ventana dinámica asociada a cada uno de sus interfaces hijo, y ésta es utilizada para controlar el flujo de datos hacia los miembros accesibles a través del correspondiente interfaz.

Dada una ventana asociada a un interfaz *i* en el sistema retransmisor *S*, el funcionamiento será el siguiente: inicialmente *S* establece un tamaño relativamente alto para dicha ventana, posteriormente este tamaño se mantendrá igual o se decrementará para ir ajustándose a las necesidades de los miembros accesibles a través de *i*. *S* detecta que debe reducir el tamaño de la ventana asociada a *i*, cuando en este interfaz se producen un número muy alto de retransmisiones. Las retransmisiones son una señal implícita de que uno o más miembros accesibles por dicho interfaz, están recibiendo más paquetes de los que son capaces de procesar. De esto modo, el sistema va evolucionando hasta alcanzar un punto de equilibrio, en el cual el tamaño de cada una de las ventanas es el adecuado como para no saturar a los miembros accesibles por dicho interfaz.

La fuente además de mantener una ventana por cada interfaz hijo, mantiene una *ventana general*. Esta ventana es estática estando su tamaño definido por el parámetro de configuración denominado *tamaño de la ventana general* (W_G). El límite inferior de esta ventana será el mínimo de los límites inferiores de las ventanas asociadas a los interfaces hijo en la fuente y el límite superior será el máximo de los límites superiores de las ventanas asociadas a los interfaces hijo en la fuente.

En RMNP existe la posibilidad de que la aplicación especifique un caudal mínimo que debe ser cursado. En el caso de que la fuente detecte que, al evolucionar el sistema para adaptarse a las necesidades de los miembros y decrementarse el tamaño de las ventanas, es imposible cursar el caudal mínimo, la fuente envía una indicación al resto de los sistemas retransmisores avisando de que no se puede reducir más, o incluso que es necesario au-

mentar, el tamaño de las ventanas. Después de recibida una de las mencionadas indicaciones, si un sistema retransmisor detecta que determinados miembros no son capaces de cursar el caudal (están originando un gran número de retransmisiones), éstos son expulsados del árbol RMNP, en lugar de reducir la ventana.

RMNP asume que el nivel inferior, distribución no fiable de datagramas multipunto, incorpora mecanismos de control de congestión. Y entre otros, algún mecanismo que evite la congestión, tomando acciones correctivas antes de que la red comience a descartar paquetes.

Una vez vistos los aspectos más relevantes del RMNP, en la próxima sección y siguientes, se pasará a describir de forma detallada el RMNP. Con el fin de clarificar la exposición, el funcionamiento del protocolo se ha subdividido en funciones y éstas se han agrupado en grandes bloques. En primer lugar, se verán las funciones relacionadas con la *distribución de mensajes multipunto*. Seguidamente, se analizan las funcionalidades relacionadas con los *procesos de confirmación y retransmisión de paquetes*. Posteriormente, se describen los procesos de *conexión, desconexión y reinicio*. Tras esto, se expone el esquema de *control de flujo* incorporado al RMNP. Y por último, se analizan los mecanismos de ajuste de los distintos *temporizadores*.

5.4 Distribución de mensajes multipunto

En esta sección se verá el proceso básico seguido por un mensaje multipunto, desde que el usuario del servicio RMNP entrega dicho mensaje en la fuente a la entidad RMNP, hasta que éste es entregado en los miembros al usuario del servicio RMNP.

Los caminos que sigue un paquete multipunto, desde que es liberado en la fuente hasta que alcanza a los miembros, vienen determinados por el nivel de encaminamiento multipunto. En concreto, el paquete sigue el árbol de distribución definido por dicho nivel de encaminamiento multipunto.

5.4.1 Distribución básica de paquetes

Con cada mensaje multipunto recibido del nivel superior, la fuente construye uno o varios paquetes de datos (también llamados paquetes DAT); este proceso de segmentación

se analiza en la sección 5.4.4. Con cada paquete DAT generado, la **fuente** realiza las siguientes operaciones:

- ❒ Asigna al paquete el siguiente número de secuencia.
- ❒ Almacena el paquete por si acaso necesita hacer retransmisiones posteriores.
- ❒ Para cada interfaz hijo *i* hace lo siguiente:
  - ♦ Si la ventana asociada a *i* ha alcanzado su tamaño máximo, retarda el envío del paquete por dicho interfaz hasta que desaparezca esta condición.
  - ♦ Si la ventana asociada a *i* no ha alcanzado su tamaño máximo:
    - Distribuye el paquete a todos sus hijos en el árbol de distribución que están accesibles por el interfaz *i*, para lo cual utiliza las funciones del servicio de distribución de datagramas multipunto y del nivel de encaminamiento multipunto.
    - Desplaza el límite superior de la ventana asociada a *i*.
    - Inicia el temporizador de retransmisión.

Este paquete liberado por la fuente llegará a, y será reenviado por, todos los encaminadores RMNP y no-RMNP pertenecientes al árbol de distribución hasta llegar a los miembros.

Un **encaminador no-RMNP** reenviará este paquete a sus hijos en el árbol de distribución, sin ser consciente de que éste incluye un paquete RMNP.

Los **encaminadores RMNP y los miembros** deben discernir, cuando reciben un paquete DAT, si éste es:

1. El paquete *esperado*. El siguiente paquete en secuencia.
2. Un paquete *fuera de secuencia*. Un paquete que no ha recibido previamente pero que se salta la secuencia de paquetes recibidos.
3. Un paquete *duplicado no confirmado*. Un paquete que ha recibido antes pero que no ha sido confirmado a su padre RMNP.

4. Un paquete *duplicado ya confirmado.* Un paquete que ha recibido antes y que ya ha sido confirmado a su padre RMNP.

5. Un paquete generado por un *error de protocolo.*

Con el fin de realizar esta distinción (ver Apéndice B, página 228), los mencionados sistemas necesitan conocer la siguiente información de estado:

- ❒ El número de secuencia del siguiente paquete esperado.

- ❒ El número de secuencia del último paquete que ha confirmado a su padre RMNP. Nótese que en la fuente no procede.

- ❒ El tamaño de la ventana general en la fuente.

Un **encaminador RMNP** actuará de forma distinta según sea la categoría del paquete recibido:

1. Si es el paquete *esperado,* lo acepta y lo reenvía a sus hijos en el árbol de distribución.

   Como ya se ha comentado y se expone de forma detallada en la sección 5.9.1, los sistemas retransmisores disponen de una ventana de transmisión asociada a cada interfaz. De aquí que en el caso de un SR, antes de distribuir el paquete por cada interfaz comprueba que la ventana asociada no ha alcanzado el tamaño máximo, en cuyo caso pospone su envío, y tras enviar el paquete actualiza el límite superior de la ventana y activa el correspondiente temporizador de retransmisión.

2. Si el paquete recibido está *fuera de secuencia,* actúa según lo expuesto en la sección 5.4.2. En dicha sección se analiza por qué todos los encaminadores RMNP, realizan la función de control intermedio de secuencia, y por tanto, sólo reenvían a sus hijos paquetes en secuencia; exponiéndose el proceso seguido.

3. Si es un paquete *duplicado no confirmado,* lo reenvía por aquellos interfaces por los cuales están accesibles hijos RMNP que no le han confirmado previamente dicho paquete. La llegada de estos paquetes es causada típicamente por el vencimiento de un temporizador de retransmisión. Este proceso de filtrado se denomina **filtrado de retransmisiones por temporizador** (ver sección 5.6.4.1).

Como se verá al analizar la función de almacenamiento, un SR controla sus propios temporizadores de retransmisión. Debido a esto, cuando un SR recibe un duplicado no confirmado, simplemente lo descarta.

4. Si es un paquete *duplicado ya confirmado* lo descarta y vuelve a enviar la última confirmación.

Un **miembro** sólo acepta paquetes en secuencia. Las razones de esta decisión de diseño se exponen en la sección 5.6. Ante la recepción de un paquete, un miembro actuará de forma distinta según sea la categoría del paquete recibido:

1. Si es el paquete *esperado,* lo acepta. En el caso de que dicho paquete no sea un fragmento de un mensaje se lo entrega al usuario del servicio; en caso contrario, completa el proceso de reensamblado del correspondiente mensaje antes de entregárselo al usuario del servicio. La función de reensamblado se expone en detalle en la sección 5.4.4.

2. Si es un paquete *duplicado ya confirmado,* lo descarta y vuelve a confirmar el último paquete ya asentido.

3. Si estaba funcionando normalmente y recibe un paquete *fuera de secuencia* hace lo siguiente:

   - ❒ Almacena de forma transitoria el paquete marcándolo como pendiente de ser aceptado.

   - ❒ Activa el temporizador de *fuera de secuencia.*

   Mientras tiene activo el temporizador de fuera de secuencia pueden ocurrir distintas cosas:

   - ❒ Que reciba otro paquete fuera de secuencia, en cuyo caso también lo almacena transitoriamente marcándolo como pendiente de ser aceptado.

   - ❒ Que reciba el paquete esperado, en cuyo caso acepta. Si dicho paquete no es un fragmento de un mensaje se lo entrega al usuario del servicio; en caso contrario, completa el proceso de reensamblado del correspondiente mensaje antes de entregárselo. Seguidamente comprueba si tenía almacenados como pendientes de ser aceptados, los paquetes siguientes al esperado, en cuyo ca-

so también los acepta y, se los entrega al usuario del servicio o completa el reensamblado. Tras esto:

- ♦ Si no tiene más paquetes pendientes de ser aceptados, detiene el temporizador.

- ♦ Si todavía tiene más paquetes pendientes de ser aceptados, no detiene el temporizador. Esto lo hace por si acaso recibe el resto de paquetes que le faltan para completar la secuencia antes de que expire el temporizador.

❒ Que el temporizador expire. En este caso, primero descarta los paquetes fuera de secuencia almacenados transitoriamente (los pendientes de ser aceptados). Y segundo, envía un asentimiento negativo (de aquí en adelante NAK) a su padre RMNP, solicitando la retransmisión del próximo paquete en secuencia. Esta función de generación de NAKs se analiza en detalle en la sección 5.6.1.

La principal razón que justifica la existencia del temporizador de *fuera de secuencia*, aunque posteriormente se analizará otra, es que las subredes atravesadas en el camino desde el padre RMNP pueden desordenar paquetes y, por lo tanto, la llegada de un paquete fuera de secuencia no implica que el/los anterior(es) se hayan perdido. De aquí, que en estas circunstancias sea conveniente retardar la generación de un NAK.

5.4.2 Control intermedio de secuencia

Los encaminadores RMNP sólo distribuyen a sus hijos paquetes en secuencia. De forma que, si en un determinado momento un encaminador RMNP estaba funcionando normalmente y detecta un salto en la secuencia hace lo siguiente:

❒ Almacena de forma transitoria el paquete marcándolo como pendiente de ser aceptado.

❒ Activa el temporizador de *fuera de secuencia.*

Mientras tiene activo el temporizador de fuera de secuencia pueden ocurrir distintas cosas:

- ❒ Que reciba otro paquete fuera de secuencia, en cuyo caso también lo almacena transitoriamente marcándolo como pendiente de ser aceptado.

- ❒ Que reciba el paquete esperado, en cuyo caso lo acepta y lo reenvía a sus hijos. Seguidamente comprueba si tenía almacenados como pendientes de ser aceptados, los paquetes siguientes al esperado, en cuyo caso también los acepta y los reenvía a sus hijos. Tras esto:

  - ♦ Si no tiene más paquetes pendientes de ser aceptados, detiene el temporizador.

  - ♦ Si todavía tiene más paquetes pendientes de ser aceptados, no detiene el temporizador. Esto lo hace por si acaso recibe el resto de paquetes que le faltan para completar la secuencia antes de que expire el temporizador.

- ❒ Que el temporizador expire. En este caso, primero descarta los paquetes fuera de secuencia almacenados transitoriamente (los pendientes de ser aceptados). Y segundo, envía un NAK a su padre RMNP, solicitando la retransmisión del próximo paquete en secuencia.

La principal razón de utilizar en los encaminadores el temporizador de *fuera de secuencia,* es la misma que se ha visto para el caso de los miembros.

El hecho de que los encaminadores RMNP incluyan la función de control intermedio de secuencia tiene dos ventajas fundamentales: en primer lugar, dado que los miembros sólo aceptan paquetes en secuencia, si los encaminadores distribuyeran paquetes fuera de secuencia estarían transmitiendo paquetes que con una probabilidad muy alta, tendrían que volver a retransmitir posteriormente. Estas retransmisiones innecesarias suponen un desperdicio de ancho de banda y tiempo de procesamiento. En segundo lugar, gracias a esta función es posible detectar antes la pérdida de paquetes (si no se incluyera, sólo los miembros detectarían la pérdida de paquetes y lo harían posteriormente), y consecuentemente, adelantar la generación del NAK correspondiente y el proceso de retransmisión asociado. En general, este adelanto de los procesos de retransmisión tiene como consecuencia una disminución del retardo medio de distribución, que es uno de los objetivos de diseño. Como contrapartida, el incluir esta función de control intermedio de secuencia supone aumentar los recursos en memoria y proceso necesarios en los encaminadores.

5.4.3 Almacenamiento intermedio

Los SRs también almacenan los paquetes pendientes de confirmación. Esto obliga a que si reciben un paquete en secuencia además de reenviarlo por los distintos interfaces hijo, lo almacenan y activan los correspondientes temporizadores de retransmisión (uno asociado a cada interfaz hijo). Esta copia almacenada se mantendrá hasta recibir de todos sus hijos RMNP el asentimiento correspondiente. Como se verá en la sección 5.6, en el caso de que expire uno de los temporizadores asociados o de que reciba de alguno de sus hijos RMNP un asentimiento negativo, iniciará un proceso de retransmisión.

5.4.4 Segmentación y reensamblado

RMNP incorpora las funciones de segmentación y reensamblado. El mecanismo usado para realizar estas funciones es similar al utilizado en X.25 [CCI88]. Cada paquete DAT lleva un flag en la cabecera (LP) que indica, si éste es el último paquete de un mensaje (LP=1), o si es un paquete intermedio lo cual implica que el siguiente paquete en la secuencia también pertenece al mismo mensaje (LP=0). Como se puede observar, este mecanismo no impone ninguna restricción al tamaño máximo de SDU[5]. RMNP, al igual que otros protocolos propuestos recientemente tal como IPv6[6] [DH95], sólo permite que la función de segmentación sea realizada en la fuente, mientras que el reensamblado se hace únicamente en destino, en este caso en los miembros.

Dado que la segmentación se realiza únicamente en la fuente, ésta debe incorporar un procedimiento que le permita calcular o estimar un valor válido para la **MTU[7] mínima en el árbol de distribución** (de aquí en adelante llamada MTTU). Este procedimiento será explicado posteriormente y por ahora simplemente se supondrá su existencia. Seguidamente se describirán de forma más detallada los procedimientos de segmentación y reensamblado RMNP:

- **Procedimiento de segmentación.** Cuando la entidad RMNP en la fuente recibe un mensaje del usuario del servicio, divide éste en paquetes de tamaño igual o menor al valor actual de MTTU y asigna a dichos paquetes números de secuencia consecutivos, activando el flag LP del último paquete de la secuencia.

[5] *Service Data Unit.*

[6] *Internet Protocol version 6.*

[7] *Maximun Transfer Unit.*

❒ **Procedimiento de reensamblado.** Cuando una entidad RMNP en un miembro recibe un paquete en secuencia y lo acepta, comprueba el valor del flag LP y hace lo siguiente:

- ♦ Si LP=0 lo almacena marcándolo como *pendiente de entrega.*
- ♦ Si LP=1 y no tiene ningún paquete como pendiente de entrega, lo cual significa que no se ha producido segmentación del mensaje original, se lo entrega al usuario del servicio. Si por el contrario tiene paquetes marcados como pendientes de entrega, reconstruye, con éstos y con el recibido, el mensaje original antes de entregárselo al usuario del servicio.

Como se puede observar, el coste introducido por la segmentación y el reensamblado en términos de proceso y ancho de banda es bajo. El proceso de construcción de los paquetes a partir del mensaje original es muy sencillo, y la pérdida de un paquete no obliga a reenviar el mensaje original (como ocurre en otros protocolos, como por ejemplo IP) sino únicamente el paquete afectado. Estas ventajas se derivan del hecho de que los paquetes RMNP incluyen un número de secuencia.

Una vez analizadas la segmentación y el reensamblado, se verá cómo la fuente puede **calcular o estimar un valor válido para la MTTU.** Existen dos posibilidades que deben ser analizadas: una primera opción es estimar un cierto valor y asignárselo de forma fija a la MTTU, y una segunda opción es incorporar un procedimiento que calcule dinámicamente un valor para la MTTU.

En primer lugar se analiza la opción de *estimar un valor para la MTTU y asignárselo de forma fija.* Normalmente los protocolos de interred no fiables, sugieren un valor de MTU mínima que deben ser capaces de transportar todas las subredes físicas que componen la interred (por ejemplo, 576 octetos en IPv4[8]). Dado esto, una opción es asignar de forma fija a la MTTU dicho valor, restando las correspondientes cabeceras. Esta opción ya ha sido analizada e igualmente descartada en otras propuestas [MD90]. A pesar de su gran sencillez, esta alternativa ha sido descartada al plantear los siguientes problemas: en primer lugar, no se puede garantizar que el valor estimado sea válido; esto se debe a que dicho valor es únicamente un valor recomendado, y por tanto existe la posibilidad de que alguna de las subredes en el árbol tenga una MTU menor a la recomendada. Y en segundo lugar, aún más importante, puede dar como resultado una pérdida en la eficiencia; debe notarse

[8] IPv4 (*Internet Protocol version 4*) sugiere una MTU mínima de 576 octetos y obliga a una MTU mínima de 64 octetos [Pos81a].

que en el caso de que todas las subredes que pertenecen al árbol de distribución tengan una MTU mucho mayor a la recomendada, se estarán generando y enviando paquetes de un tamaño menor al necesario. El mandar paquetes más pequeños a la MTU mínima en el árbol real, provoca que no se haga un uso eficiente de los recursos de la interred y se obtenga un caudal subóptimo.

Seguidamente, se analiza la posibilidad de *calcular dinámicamente* el valor que debe ser asignado a la MTTU. Lo normal es que este procedimiento sea ejecutado al inicio de la fase de establecimiento de la conexión aunque como se verá posteriormente, en casos excepcionales, puede ejecutarse en la fase de transferencia de datos. RMNP propone dos alternativas al procedimiento de cálculo dinámico de la MTTU. El uso de una u otra dependerá del protocolo multipunto no fiable sobre el que esté implementado el RMNP.

1. Si el nivel inferior incluye un procedimiento para averiguar la MTU mínima entre el origen y los destinos, como es el caso del IPv6, entonces dicho procedimiento es usado para calcular dinámicamente un valor a la MTTU.

2. En el caso de que el nivel inferior no incluya dicho procedimiento, como por ejemplo IPv4, entonces RMNP utilizará un algoritmo de **reducción exponencial binaria.** Este algoritmo, detallado en la Figura 5.5, progresa en ciclos. En cada ciclo la fuente asigna un valor a la MTTU e intenta averiguar si éste valor es válido. Si comprueba que el valor asignado a la MTTU no es válido, en el ciclo siguiente le asigna un valor la mitad del anterior. Para comprobar si es válido se utilizan dos paquetes de control: el paquete de descubrimiento de la MTTU (llamado MTTUD) y su confirmación correspondiente (llamada MTTUD_ACK). Cada uno de estos paquetes incluye en el campo *Identificador del ciclo de descubrimiento*, el identificador del ciclo en el que fue generado el paquete correspondiente. Este identificador permite a los distintos sistemas asociar cada MTTUD_ACK con su MTTUD correspondiente. En cada ciclo la fuente envía N_{MTTUD}[9] paquetes MTTUD solicitando al nivel inferior que no realice la función de segmentación (de otro modo el valor calculado no sería real), y espera a recibir el correspondiente MTTUD_ACK. Si este asentimiento llega, indica que *todos* los miembros han recibido una copia del paquete MTTUD, y han generado el correspondiente MTTUD_ACK, con lo cual el valor actual asignado a MTTU es válido. Es preciso enviar N_{MTTUD} paquetes MTTUD en lugar de uno sólo, para reducir la probabilidad de que la fuente no reciba el MTTUD_ACK correspondiente, no porque el valor asignado a la MTTU no sea válido, sino simplemente porque en algún punto del árbol se perdió el paquete MTTUD o alguno de los MTTUD_ACKs enviados por los miembros. Sin embar-

[9] Parámetro de configuración.

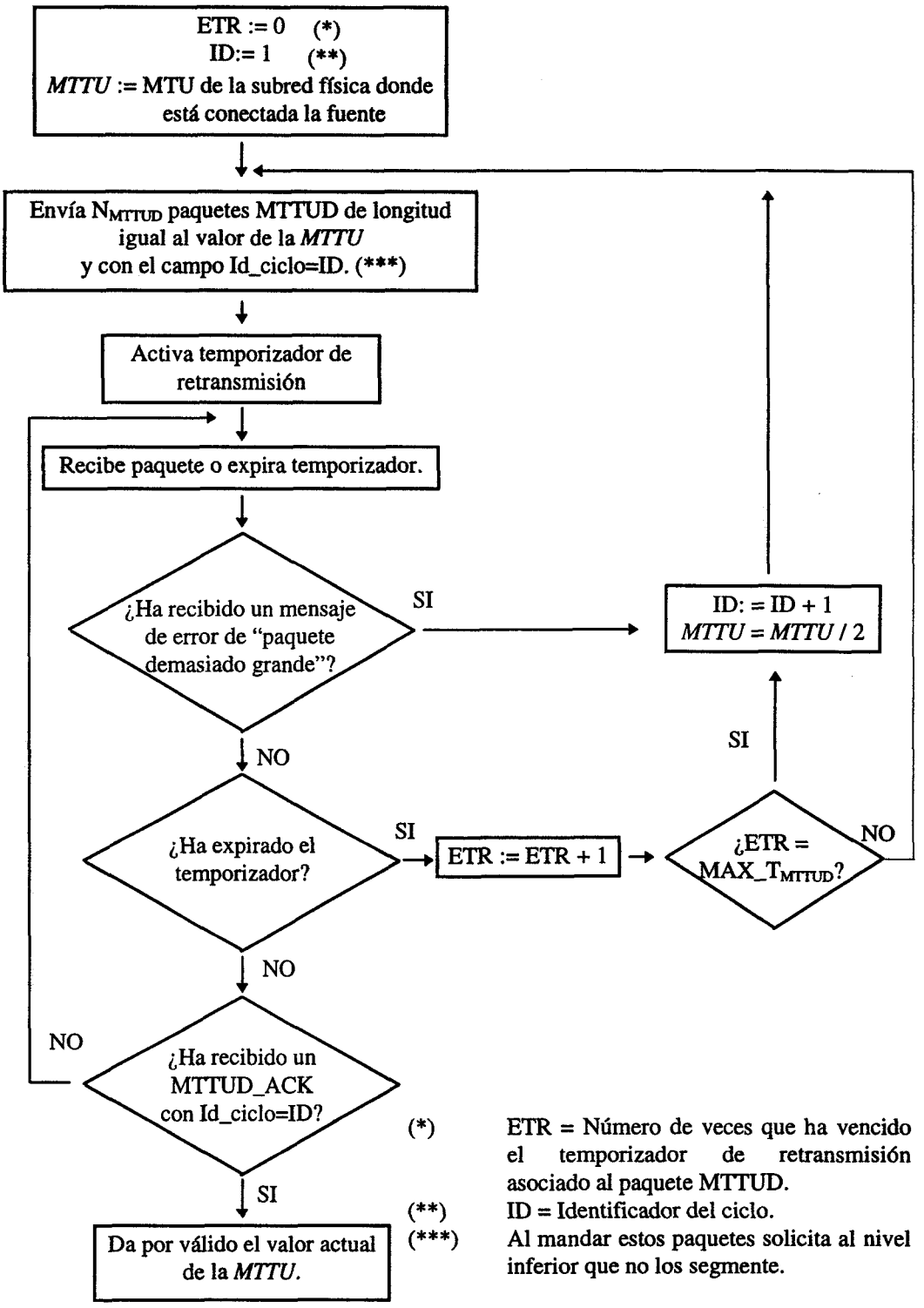


Figura 5.5: Algoritmo de reducción exponencial binaria

go, tampoco es deseable que N_{MTTUD} tome un valor demasiado alto, ya que esto podría provocar en la fuente una implosión de mensajes de error del tipo "paquete demasiado grande" (la utilidad de estos mensajes de error se comenta posteriormente).

Los encaminadores RMNP realizan un proceso de agregación con los paquetes MTTUD_ACKs similar al realizado con los ACKs (ver sección 5.5.1). Cuando un encaminador ha recibido un paquete MTTUD_ACK asociado al ciclo *i*, desde cada uno de sus hijos, envía hacia su padre un MTTUD_ACK asociado a dicho ciclo *i*. Debe notarse que cuando se manda un paquete MTTUD con un tamaño demasiado grande, es imposible que éste atraviese todas las subredes y llegue a todos los destinos. En esta situación pueden darse dos casos diferentes, según sea el servicio ofrecido por el nivel inferior, que induzcan a reducir el valor asignado a MTTU a la mitad. O bien, se recibe un mensaje de error del nivel inferior indicando que el paquete no ha podido ser entregado, ya que para alcanzar a uno o más destinos es necesario segmentar el paquete RMNP (por ejemplo, si se usa IPv4 se recibirá un mensaje ICMP de destino inalcanzable con un código que indica "se necesita hacer segmentación y el bit 'no fragmentar' está activado"), de aquí en adelante estos mensajes de error se denominan de forma genérica mensajes "paquete demasiado grande"; o bien, en el caso de que el nivel inferior no incorpore dichos mensajes de error, situación poco probable, cuando expira un cierto número de veces (MAX_T_{MTTUD}[10]) el temporizador de retransmisión asociado al paquete MTTUD, se considera que la MTTU tiene un valor demasiado alto como para llegar a todos los miembros, y se inicia un nuevo ciclo reduciéndose dicho valor a la mitad.

Como otra opción de diseño se ha considerado la posibilidad de utilizar el mismo algoritmo expuesto pero usando paquetes de datos, en lugar de incorporar paquetes de control adicionales. En esta solución se pueden plantear situaciones anómalas. Por ejemplo supóngase que en el primer ciclo del algoritmo el mensaje de datos a enviar va contenido en un único paquete con número de secuencia uno; y un primer miembro acepta este paquete mientras que un segundo miembro ni siquiera lo recibe, por existir en el camino entre la fuente y este miembro una o varias subredes con una MTU más restrictiva. Como es obvio, la fuente no recibe el asentimiento correspondiente y como consecuencia vuelve a reenviar el mismo mensaje, pero esta vez segmentado en dos paquetes (por ejemplo con números de secuencia 1 y 2). El primer miembro al recibir estos paquetes hace lo siguiente: el primer paquete lo desecha pero el segundo lo acepta, sin poder detectar que el contenido de este segundo paquete ya estaba incluido en el paquete que recibió originalmente (contenía el mensaje completo). El resultado es que este primer miembro entrega datos duplicados al nivel superior.

[10] Parámetro de configuración.

El algoritmo aquí propuesto calcula un valor válido para ser usado como MTU mínima en los caminos desde la fuente hasta los miembros, pero en ningún caso como la MTU mínima en los caminos desde los miembros hasta la fuente, nótese que estos segundos caminos son únicamente utilizados por los paquetes de control que van desde los miembros hacia la fuente. Este hecho no supone ningún problema, pues dado el tamaño de los paquetes de control, éstos podrán viajar desde los miembros hasta la fuente sin ser segmentados, con la excepción de que la MTU mínima obligada en la interred fuera excesivamente pequeña, situación poco probable.

La definición actual del RMNP asume que el nivel inferior no ofrece ninguna facilidad para calcular dinámicamente la MTTU y por tanto, utiliza el algoritmo de reducción exponencial binaria. Asimismo asume que dicho nivel inferior incluye mensajes de error del tipo "paquete demasiado grande".

Seguidamente se pasa a analizar los problemas surgidos cuando el nivel inferior también realiza las funciones de segmentación y reensamblado. Obviamente, a parte de la segmentación y reensamblado al nivel RMNP, el nivel inferior también puede realizar estas funciones. En concreto, algunos protocolos multipunto de red no fiables realizan estas funciones y además, con la particularidad de que únicamente se reensambla en destino, esto tiene como consecuencia, que si un paquete RMNP es segmentado por el nivel inferior en varias PDUs y no se reensambla en los encaminadores, la entidad RMNP no podrá analizar los paquetes RMNP completos. Sin embargo, con el fin de hacer posibles algunas de las funciones, como por ejemplo el control intermedio de secuencia, es preciso que las entidades RMNP en los encaminadores analicen todos los paquetes RMNP completos. Para resolver esta problemática se plantean dos soluciones dependiendo de como sea el servicio ofrecido por el nivel inferior:

1. Si es posible solicitar al nivel inferior que no haga segmentación (por ejemplo, en IPv4 usando el bit "no fragmentar"[11]) entonces RMNP solicita este servicio. Esto significa que siempre que sea posible, esto es, si no hay cambios en la MTTU mientras hay paquetes pendientes de confirmación, *las funciones de segmentación y reensamblado se realizan únicamente a nivel RMNP*. Debe comentarse que RMNP no es un caso excepcional al precisar que el nivel inferior no segmente, para más detalles consultar [KM87].

[11] Conocido como bit *don't fragment*.

2. Si no es posible solicitar al nivel inferior que no haga segmentación entonces, además de en los destinos, también en los encaminadores RMNP se realiza la función de reensamblado a dicho nivel inferior. Como es obvio, el incluir esta función en los encaminadores supone un gasto en proceso y en memoria. Sin embargo, no es preciso introducir ningún mecanismo adicional para forzar a los paquetes correspondientes a un mismo mensaje a atravesar los mismos encaminadores, ya que por la propia naturaleza del problema del encaminamiento multipunto todos los paquetes se propagan siguiendo el mismo árbol de distribución, y un encaminador RMNP recibirá *todos* los paquetes correspondientes a un mensaje.

Dado que la mayoría de los actuales protocolos multipunto no fiables, permiten solicitar que no se realice la función de segmentación, en la definición actual de RMNP únicamente se contempla la opción 1 y siempre que sea posible las funciones de segmentación y reensamblado se realizan únicamente a nivel RMNP.

Normalmente, la MTTU se mantendrá constante durante todo el tiempo que dure la conexión. Sin embargo, puede variar con la entrada o salida de nuevos miembros en el grupo o con un cambio en el árbol de distribución. Nótese que la probabilidad de que varíe la MTTU aumenta, si las MTUs de las subredes que componen la interred son muy dispares. En el caso de que la MTTU real aumente y no se recalcule un nuevo valor para la variable asociada en la fuente, surge un problema de pérdida de eficiencia. Para solventar este problema de pérdida de eficiencia es necesario obligar a la fuente a que periódicamente recalcule dinámicamente un valor para la MTTU (para un análisis más detallado de este problema ver [MD90]). Esta posibilidad de recálculo periódico de la MTTU no ha sido contemplada en la definición actual del RMNP, esto es debido a que la situación expuesta no será muy frecuente y aunque causa una pérdida en la eficiencia no impide que todos los miembros continúen recibiendo todos los paquetes RMNP. En cambio, si disminuye la MTTU real y no se recalcula un nuevo valor para la variable asociada, al menos un miembro quedará inaccesible, esto es, no podrá recibir más paquetes.

La situación es más compleja de lo que podría parecer a primera vista ya que, cuando se numeran los paquetes y se utilizan números de secuencia con un significado global, como es el caso del RMNP, si existen paquetes pendientes de confirmación, no es posible recalcular el valor de la MTTU garantizando que no se producirán situaciones anómalas. Utilizar números de secuencia con un significado global significa que el paquete conocido como paquete 3 de la ventana actual es el mismo para todos los sistemas. Dichas situaciones anómalas son similares a las expuestas para el caso de que se utilicen paquetes de datos para calcular la MTTU.

A continuación se razonan las anteriores afirmaciones con un ejemplo. Supóngase que cuando la fuente detecta que ha cambiado la MTTU tiene dos paquetes pendientes de confirmación, con números de secuencia 30 y 31, siendo ambos el resultado de segmentar un mensaje (m_x). En ese momento, la situación de dichos paquetes en los distintos miembros es muy dispar. Un miembro, m_1, ha entregado dichos paquetes al usuario del servicio y está esperando el paquete 32; mientras que otro miembro, m_2, que tiene en su camino desde la fuente la subred o subredes que están imponiendo la limitación a la MTTU, no ha recibido dichos paquetes y aún está esperando el 30. En estas circunstancias, si la fuente reduce a la mitad la MTTU, vuelve a segmentar m_x obteniendo como resultado cuatro paquetes con números de secuencia 30, 31, 32, 33, y los reenvía, m_1 aceptará los paquetes 32 y 33 con lo cual el usuario del servicio recibirá información duplicada; y si por el contrario no vuelve a segmentar el mensaje con un valor de MTTU más pequeño, m_2 no puede recibir el mensaje.

Después de analizada esta problemática, se concluye que el comportamiento del RMNP al disminuir la MTTU será el siguiente: si la fuente recibe un mensaje de error del tipo "paquete demasiado grande" indicando que un paquete no ha podido ser entregado, dado que para alcanzar a uno o más destinos es necesario segmentar el paquete RMNP, ésta actúa de un modo u otro dependiendo si tiene o no paquetes pendientes de confirmación. Si no tiene paquetes pendientes de confirmación, vuelve a calcular un valor para la MTTU. Si por el contrario existen paquetes pendientes de confirmación, la fuente detiene temporalmente el envío de nuevos paquetes y reenvía todos los paquetes pendientes de confirmación, pero esta vez sin solicitar al nivel inferior que no segmente, o lo que es lo mismo permite al nivel inferior que segmente. Cuando ha recibido confirmación a todos los paquetes que estaban pendientes de confirmación, vuelve a recalcular la MTTU y continúa con el proceso normal enviando nuevos paquetes y solicitando al nivel inferior que no segmente. Como es obvio esta solución obliga a que los encaminadores RMNP incluyan al nivel inferior la función de reensamblado.

Aunque es muy poco probable, puede ocurrir que el nivel inferior en determinadas circunstancias no genere dichos mensajes del tipo "paquete demasiado grande" (un ejemplo de esta situación es expuesto en [Kno93]). En este caso, cuando se solicita a dicho nivel inferior que no segmente, y un determinado encaminador encuentra imposible hacer la entrega de un determinado paquete sin segmentarlo, el paquete es descartado, pero no se genera ningún mensaje de error para avisar a la fuente. Esto puede resolverse haciendo que cuando el temporizador de retransmisión de un paquete ha expirado un cierto número de veces, la fuente sospeche que puede ser debido a un cambio en la MTTU y actúe del mismo modo a cuando recibe un mensaje del tipo "paquete demasiado grande". Como ya se

ha comentado la definición actual de RMNP asume que el nivel inferior, siempre que es preciso, genera mensajes del tipo "paquete demasiado grande".

Además de la solución aquí propuesta, también ha sido considerada una alternativa de diseño qué no plantea problemas al cambiar la MTTU en la mitad de una transferencia. Consiste en numerar los octetos enviados, en vez de numerar los paquetes (esta misma aproximación es seguida por otros protocolos, ampliamente utilizados, como por ejemplo TCP). El numerar los octetos obliga a que los paquetes de datos incluyan un campo con el número de secuencia del primer octeto contenido en el campo de datos de usuario, y un campo longitud que indica o permite averiguar la longitud del campo datos de usuario.

Planteada esta alternativa deben analizarse los siguientes aspectos:

- Es posible utilizar los mismos algoritmos de segmentación y reensamblado ya vistos en esta sección.

- Es posible incluir las funciones de segmentación y reensamblado en los encaminadores sin que surjan problemas de entrega de duplicados. La segmentación se haría de forma que: (1) de un paquete con el flag LP desactivado se obtendrían dos o más paquetes con el flag LP desactivado, y (2) de un paquete con el flag LP desactivado se obtendrían *n* paquetes, los *n-1* primeros con el flag LP desactivado y el último con este flag activado. El proceso de reensamblado en los encaminadores sería tal que combinaría en un único paquete dos o más paquetes que incluyan octetos en secuencia, con la única restricción de que nunca se combinarían paquetes que originalmente pertenecieran a SDUs distintas. El flag LP de este paquete surgido de la combinación de *n* consecutivos, iría activado o no, dependiendo de si estaba o no activado en el último de los *n* paquetes

- Un cambio en la MTTU cuando hay paquetes pendientes de confirmación, no plantea problemas de entrega de datos duplicados. La solución sería que cuando se detectara un problema de este tipo, se volverían a segmentar paquetes previamente segmentados, e incluso esta segmentación podría hacerse en el encaminador que detectara el cambio en la MTTU, sin ser preciso avisar a la fuente y que fuera ésta la que resolviera el problema.

Las ventajas de este planteamiento son las siguientes:

- Permite que en distintos encaminadores puedan usarse para segmentar, distintos valores de MTU mínima, dependiendo de las características de las subredes atrave-

sadas entre un encaminador RMNP y el siguiente. Esto permite un aprovechamiento más eficiente de los recursos de red.

- No plantea ningún problema cuando cambia la MTTU y hay paquetes pendientes de confirmar.

- En ningún caso es necesario permitir al nivel inferior que segmente, con lo cual no es obligatorio que exista la función de reensamblado a nivel inferior en los encaminadores RMNP.

Sin embargo, también plantea los siguientes inconvenientes:

- El hecho de numerar octetos en lugar de paquetes obliga a que los distintos campos que incluyen números de secuencia sean más largos. Esto provoca un aumento en la longitud de la cabecera de los distintos paquetes con el consiguiente desperdicio en ancho de banda.

- Los encaminadores deben incluir un procedimiento que les permita calcular de forma dinámica cual es la mínima MTU en el camino hasta cada uno de sus hijos RMNP.

- La lógica del protocolo es más compleja. Nótese que este planteamiento hace más complejas, entre otras, las operaciones de agregación de ACKs y filtrado de NAKs.

- Es necesario realizar más proceso en los encaminadores.

Este planteamiento fue descartado después de analizar los pros y los contras anteriormente expuestos. De modo que, como se ha visto, RMNP numera los paquetes y la función de segmentación se realiza únicamente en la fuente, mientras que la función de reensamblado se realiza únicamente en los miembros.

5.5 Confirmación de paquetes de datos

La confirmación de los paquetes DAT recibidos correctamente se hace con paquetes ACK. Un único ACK sirve para la confirmación de varios DATs (asentimientos acumulativos). La recepción en un nodo RMNP de un ACK_*n* (paquete ACK que contiene en el campo número de secuencia el valor *n*) enviado por un hijo *i*, confirma que todos los des-

cendientes RMNP, accesibles a través de dicho hijo *i*, han recibido correctamente hasta el DAT_*n* incluido éste.

El proceso de confirmación se inicia en los miembros del grupo y va propagándose por el árbol RMNP hacia la fuente.

Un **miembro** cuando recibe el paquete esperado con número de secuencia *n* y lo acepta, genera un ACK_*n*.

En el transcurso de la propagación de los ACKs por el árbol RMNP hacia la fuente, todos los **encaminadores RMNP** realizan un proceso de *agregación de ACKs* que se describe en la próxima sección.

Los **sistemas retransmisores** (fuente y SRs) cuando han recibido el asentimiento correspondiente a un paquete, desde todos sus hijos RMNP asociados al interfaz *i*, desplazan el límite inferior de la ventana asociada a dicho interfaz y detienen el temporizador asociado al mencionado paquete en dicho interfaz. Asimismo cuando han recibido el asentimiento correspondiente a un paquete desde *todos* sus hijos RMNP, liberan el buffer asociado a dicho paquete.

5.5.1 Agregación de ACKs

Cada encaminador RMNP para realizar la función de agregación de ACKs, utiliza la siguiente información de estado: cual ha sido el último paquete que él mismo ha asentido a su padre RMNP, y hasta qué paquete le ha asentido cada uno de sus hijos RMNP. Esta información es actualizada cada vez que un ACK es recibido o enviado.

El proceso de agregación de ACKs consiste en lo siguiente: cuando un encaminador RMNP recibe un ACK que confirma hasta un paquete con número de secuencia igual o mayor a un número de secuencia previamente confirmado por el resto de sus hijos RMNP, entonces el encaminador envía a su padre RMNP, un ACK con el mayor número de secuencia ya confirmado por todos sus hijos RMNP.

5.6 Recuperación de paquetes perdidos

En esta sección se verá el proceso de recuperación de los paquetes perdidos. En RMNP la retransmisión de dichos paquetes se hace preferentemente *desde el SR* más cercano al punto donde se originó el problema o en su defecto *desde la fuente*. Por SR más cercano a un punto se entiende, aquel SR que está a menor número de saltos en el camino desde dicho punto a la fuente, estando determinado dicho camino por el árbol RMNP.

Un proceso de retransmisión se inicia, o bien por el vencimiento de un temporizador de retransmisión (en la fuente o en un SR), o bien porque uno o varios sistemas (miembros o encaminadores RMNP) así lo han solicitado explícitamente utilizando un NAK. Dada la pérdida de un paquete de datos, el NAK es generado por el primer sistema (miembro o encaminador), que detecta un salto en la secuencia. Dicho NAK es enviado hacia la fuente a través del árbol RMNP (si procede cada nodo RMNP envía el NAK a su padre RMNP), y la solicitud de retransmisión se satisface a partir del primer sistema, en dicho camino hacia la fuente, que realiza las funciones de almacenamiento y retransmisión.

En el proceso de recuperación de paquetes perdidos, los encaminadores RMNP realizan distintos procesos de filtrado que tienen por finalidad optimizar la utilización de los recursos de la interred.

En las siguientes subsecciones se analizan en detalle las funciones realizadas en los sistemas RMNP para lograr la recuperación de paquetes. Para realizar dichas funciones estos sistemas utilizan la siguiente información estado:

- A partir de qué paquete está pendiente de volver a recibir desde su padre RMNP[12] (petición realizada mediante un NAK). Esta información se actualiza cuando se envía un NAK y cuando se reciben los paquetes de datos solicitados.

- A partir de qué paquete le ha solicitado retransmisión cada uno de sus hijos RMNP[13], estando dicha petición pendiente de satisfacer. Esta información se actualiza, si procede, cuando se recibe un NAK o un ACK, y cuando se reenvían dichos paquetes de datos.

- Número de secuencia en el qué quedó interrumpida la distribución básica de paquetes, cuando se inició el proceso de recuperación.

[12] Nótese que no procede en el caso de la fuente, ni en el de los SRs.

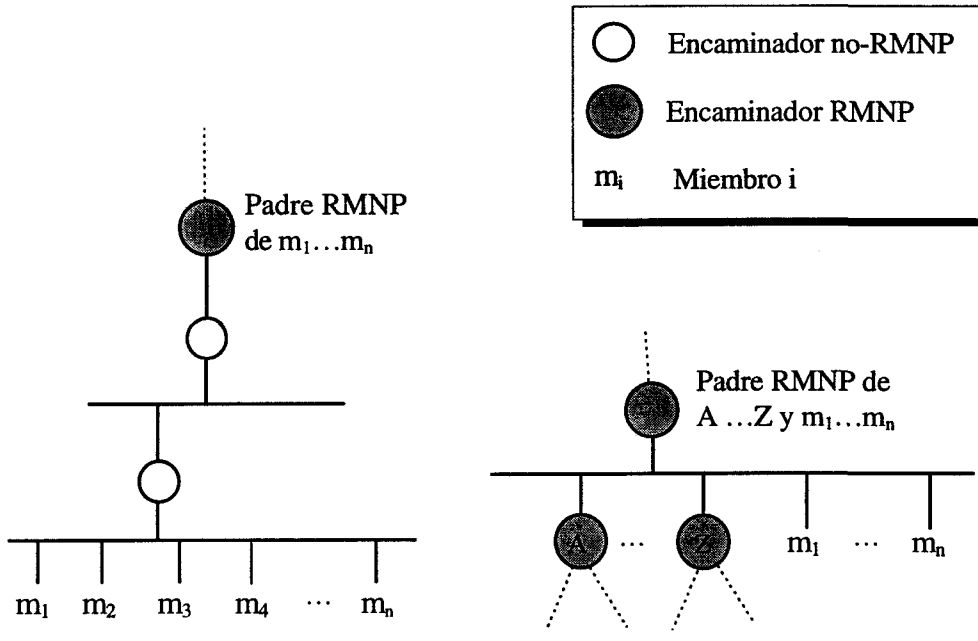
[13] Nótese que no procede en el caso de los miembros.

- El número de secuencia del último paquete que el sistema ha confirmado a su padre RMNP. Esta información es actualizada cuando se envía un ACK.

- El número de secuencia del último paquete que le ha confirmado cada uno de sus hijos RMNP. Esta información es actualizada cuando se recibe un ACK.

5.6.1 Generación de NAKs

Los NAKs son generados en los miembros y en los encaminadores RMNP cuando detectan un salto en la secuencia, de acuerdo con los procedimientos vistos en las secciones 5.4.1 y 5.4.2.



a. Varios miembros en la misma RAL

b. Varios miembros y encaminadores en la misma RAL

Figura 5.6: Sistemas conectados a la misma subred

Dado que será usual que varios hijos RMNP estén conectados a una misma subred que ofrece un servicio multipunto (ver Figura 5.6), en el caso de la pérdida de un paquete, existe una probabilidad muy alta de que este problema afecte a varios e incluso a todos los hijos RMNP. De aquí que deban analizarse dos posibles soluciones a la generación de

NAKs, o bien enviar estos paquetes como punto a punto o bien como multipunto, a continuación se comentan cada una de estas alternativas:

- **NAKs como paquetes multipunto.** Cuando un hijo RMNP (miembro o encaminador) debido a un salto en la secuencia detecta que le falta el paquete *n*, hace lo siguiente:

  - Si dicho hijo no está conectado a una subred que proporciona un servicio multipunto, envía un NAK_*n* utilizando la dirección punto a punto de su padre.

  - Si dicho hijo está conectado a una subred que proporciona un servicio multipunto, espera un tiempo aleatorio antes de enviar ningún paquete NAK. En el caso de que durante dicha espera aleatoria reciba un NAK, enviado como un paquete multipunto por otro hijo RMNP en la misma subred, que también solicita la retransmisión del paquete *n*, detiene la espera aleatoria y se abstiene de enviar ningún NAK. En el caso contrario de que finalice la espera aleatoria, envía dos paquetes NAK:

    1. Un NAK con la dirección multipunto del grupo y con el alcance limitado a la subred (por ejemplo, si se utiliza IPv4 con TTL[14]=1).

    2. Un NAK con la dirección punto a punto de su padre RMNP. Este segundo NAK es preciso, ya que el padre RMNP puede no estar conectado a la misma subred que el sistema que ha enviado el NAK multipunto (ver Figura 5.6, configuración a.), en cuyo caso el padre no lo recibiría.

- **NAKs como paquetes punto a punto.** Nada más detectar un salto en la secuencia, el sistema correspondiente (miembro o encaminador RMNP) envía un paquete NAK utilizando la dirección punto a punto de su padre RMNP.

Vistas las dos alternativas, éstas serán comparadas:

- Ante la pérdida de un mismo paquete en *n* hijos RMNP conectados a la misma subred, si se utilizan NAKs multipunto el padre RMNP sólo procesa uno o dos NAKs, mientras que si se emplean NAKs punto a punto el padre RMNP se verá obligado a procesar *n* NAKs idénticos (uno por cada hijo RMNP). Incluso, en el caso extremo de que *n* tuviera un valor muy alto, podría producirse un problema

[14] *Time To Live.*

de implosión en dicho padre, aunque debe notarse que esta situación es poco probable, ya que como máximo recibirá un NAK por hijo y no será usual que un padre RMNP tenga un número de hijos RMNP excesivamente elevado.

- El empleo de NAKs multipunto implica una optimización en la utilización de los recursos de red.

- Si se generan NAKs multipunto, el padre RMNP no puede mantener información relativa a las solicitudes de retransmisión asociada a cada hijo, sino a cada interfaz. Esto impide que se tenga información precisa sobre un hijo cuando este cambia de interfaz.

- Si un *único* sistema detecta un salto en la secuencia, en el caso de que se utilice el procedimiento de NAKs multipunto se envían dos NAKs, mientras que con el procedimiento de NAKs punto a punto se envía sólo uno.

- Si se emplean NAKs multipunto se retarda durante la espera aleatoria el envío del NAK con lo cual se retarda el inicio del proceso de retransmisión. Sin embargo, dado que un valor razonable para la espera aleatoria estará en el rango [0, 5 sg.] este retardo no es excesivo. En el otro caso de utilizar NAKs punto a punto, el NAK se envía inmediatamente y por lo tanto no se retarda el proceso de retransmisión

Se puede concluir que el procedimiento de generar NAKs multipunto será tanto más conveniente cuanto mayor sea el número de hijos RMNP conectados a la misma subred que ofrece un servicio multipunto. Sin embargo, si el número de hijos RMNP conectados a la misma subred es bajo, será más conveniente generar NAKs punto a punto, esto se debe principalmente a la mayor simplicidad del proceso y a que no se retarda el proceso de retransmisión. En la definición actual de RMNP la alternativa adoptada es generar NAKs punto a punto.

5.6.2 Filtrado de NAKs

Todo BR recuerda si tiene alguna solicitud de retransmisión pendiente de satisfacer y a partir de qué paquete. Haciendo uso de esta información, cada vez que recibe un NAK desde uno de sus hijos, únicamente lo reenvía a su padre RMNP, si la solicitud corresponde a un paquete anterior al primero que él mismo ya tiene pendiente de recibir. Por ejemplo, si tiene pendiente de recibir a partir del paquete 3 y recibe un NAK_*5* desde uno de

sus hijos, filtra este NAK y no lo reenvía a su padre. Obviamente, aunque no lo reenvía a su padre sí actualiza la información de estado asociada a dicho hijo, anotando que éste desea recibir a partir del paquete 5.

Esta función añade muy poco proceso adicional en los BRs y sin embargo evita retransmisiones innecesarias. Por ejemplo, puede ocurrir que dos sistemas detecten el mismo salto en la secuencia lo que origina el envío de dos NAKs hacia la fuente. Si ambos NAKs coinciden en su camino hacia la fuente a partir de BR_i y tienen como destino SR_j, es muy probable que uno de estos NAKs sea filtrado en BR_i y se inicie un único proceso de retransmisión en SR_j, aunque naturalmente los paquetes de datos retransmitidos llegarán a los dos sistemas que detectaron el salto en la secuencia. Nótese que si no se realizara esta función de filtrado, los dos NAKs llegarían a SR_j, y en el caso de que el segundo NAK llegara muy distanciado en tiempo del primero, se originaría un segundo proceso de retransmisión innecesario.

Carece de sentido que los SRs realicen la función de filtrado de NAKs, pues ellos mismos satisfacen las retransmisiones solicitadas, con lo cual no reenvían hacia su padre los NAKs recibidos.

5.6.3 Retransmisión

La retransmisión de paquetes se realiza desde la fuente o desde un SR. Los procesos de retransmisión están asociados a interfaces hijo y pueden ser motivados por el vencimiento de un temporizador de retransmisión o por una solicitud explícita (la recepción de un NAK). Si un NAK encuentra en su camino hacia la fuente un SR, éste realizará las retransmisiones solicitadas, y en caso contrario lo hará la fuente. Con la finalidad de clarificar esta exposición ambos procesos de retransmisión se analizan de forma separada. En la sección 5.6.3.1 se expone el comportamiento asociado a un interfaz en el que se está llevando a cabo un proceso de *retransmisión por temporizador* y en la sección 5.6.3.2 se presenta el comportamiento asociado a un interfaz en el que se está llevando a cabo un proceso de *retransmisión por solicitud explícita.*

Un sistema retransmisor (SR o fuente) que no tiene ningún proceso de retransmisión asociado a ninguno de sus interfaces hijo, al vencer un temporizador de retransmisión o recibirse un NAK, interrumpe el proceso de distribución básica de paquetes, asocia al co-

rrespondiente interfaz un procedimiento de retransmisión por temporizador o por solicitud explícita respectivamente y a partir de ese instante tiene el siguiente comportamiento:

- Si *recibe un paquete de datos,* lo descarta. En el caso de que dicho paquete fuera un duplicado ya confirmado, además de descartar el paquete también reenvía a su padre RMNP el último ACK.

- Si *recibe un ACK_n* desde un hijo accesible por un interfaz que no está en proceso de retransmisión, actualiza la información de estado asociada a dicho hijo, y si procede, realiza un proceso de agregación de ACKs

- Si *recibe un NAK* desde un hijo accesible por un interfaz que no está en proceso de retransmisión, asocia a dicho interfaz un proceso de retransmisión por solicitud y consecuentemente el comportamiento para dicho interfaz será el detallado en la sección 5.6.3.2.

- Si *expira un temporizador* asociado a un interfaz que no está en proceso de retransmisión, asocia a dicho interfaz un proceso de retransmisión por temporizador y consecuentemente el comportamiento para dicho interfaz será el detallado en la sección 5.6.3.1.

Una vez finalizados los procesos de retransmisión asociados a los distintos interfaces, el sistema (SR o fuente) continuará con el proceso de distribución básica de paquetes, a partir del punto donde quedó interrumpida la secuencia.

5.6.3.1 Retransmisión por temporizador

Al expirar un temporizador de retransmisión asociado al interfaz *i*, el comportamiento para dicho interfaz será el siguiente:

- Se detienen todos los temporizadores de retransmisión asociados a dicho interfaz *i*.

- Se distribuyen nuevamente todos los paquetes pendientes de confirmación por dicho interfaz *i*. Obviamente, cada vez que se retransmite un nuevo paquete se le vuelve a asociar un temporizador de retransmisión. Como se verá en la sección 5.10.1 el temporizador de retransmisión se ajusta de forma dinámica, esto tiene como consecuencia el que puede expirar un temporizador cualquiera, y no únicamente el asociado al primer paquete pendiente de confirmación, sin embargo independientemente de a qué paquete este asociado el temporizador que ha expirado, se vuelven a retransmitir todos los paquetes pendientes de confirmación.

Mientras está en marcha el proceso de retransmisión, la actuación en el mencionado interfaz variará según distintos sucesos:

- Si *recibe un NAK desde un hijo accesible por el interfaz i*, simplemente lo descarta.

- Si *recibe un ACK_n desde un hijo accesible por el interfaz i*, actualiza la información de estado asociada a dicho hijo, y si procede, realiza un proceso de agregación de ACKs. Si después de actualizar dicha información de estado se cumple, que el próximo paquete a retransmitir es el *m* y que todos sus hijos RMNP accesibles por el interfaz *i* le han confirmado hasta el paquete *m+k*, el proceso de retransmisión continuará a partir del paquete *m+k+1*.

La retransmisión en el interfaz *i* finaliza, o bien porque se han terminado de retransmitir todos los paquetes pendientes de confirmación, o bien porque se han recibido uno o más ACKs y *todos* sus hijos accesibles por dicho interfaz le han asentido *todos* los paquetes que estaban pendientes de confirmación.

Obviamente, conviene que las *retransmisiones por temporizador* se hagan desde el sistema retransmisor (fuente o SR) más cercano al punto o puntos que precisan de la recuperación de determinados paquetes. Esto se consigue gracias al mecanismo de ajuste de temporizadores que se verá en la sección 5.10.1.

5.6.3.2 Retransmisión por solicitud explícita

En un sistema retransmisor (SR o fuente), al recibirse un NAK_*n* desde un hijo asociado al interfaz *i*, el comportamiento para dicho interfaz será el siguiente:

- Se *distribuyen nuevamente todos los paquetes pendientes de confirmación a partir del n*. Obviamente, cada vez que se retransmite un nuevo paquete se actualiza el temporizador de retransmisión asociado.

Mientras está en marcha el proceso de retransmisión, la actuación en el mencionado interfaz variará según distintos sucesos:

- Si *recibe un ACK_n desde un hijo accesible por el interfaz i*, actualiza la información de estado asociada a dicho hijo, y si procede, realiza un proceso de agregación de ACKs. Si después de actualizar dicha información de estado se cumple, que el próximo paquete a retransmitir es el *m* y que todos sus hijos RMNP accesibles por

el interfaz *i* le han confirmado hasta el paquete *m+k*, el proceso de retransmisión continuará a partir del paquete *m+k+1*.

- Si *expira un temporizador de retransmisión*, el comportamiento dependerá de a qué paquete está asociado dicho temporizador y en qué punto comenzó el proceso de retransmisión. Si se supone que dicho temporizador está asociado al paquete *n* y que el proceso de retransmisión por solicitud se ha iniciado a partir del paquete *m* (se recibió un NAK_*m*), el comportamiento será el siguiente:

  - Si $n < m$ detiene el proceso de retransmisión actual e inicia un nuevo proceso de retransmisión por temporizador comenzando con el paquete *n*.

  - Si $n \geq m$ lo ignora. Nótese que esta situación no se dará en condiciones normales, pues cada vez que se retransmite un nuevo paquete se actualiza el temporizador asociado.

- Si *recibe un NAK_n desde un hijo accesible por el interfaz i*, actualiza la información de estado asociada a dicho hijo. Obviamente, esta información de estado sólo se actualiza si dicho hijo, o bien no tenía ninguna solicitud de retransmisión pendiente, o si tenía una solicitud pendiente era a partir de un paquete mayor a *n*. El resto del comportamiento depende de como transcurre el proceso de retransmisión ya iniciado, por ejemplo si se supone que el próximo paquete a retransmitir es el paquete *m*, el comportamiento será el siguiente:

  - Si $n < m$ detiene el proceso de retransmisión actual e inicia otro proceso de retransmisión por NAKs, comenzando en el paquete *n*.

  - Si $n \geq m$ continúa con el proceso de retransmisión. En este caso se está solicitando la retransmisión de un paquete que ya está pendiente de ser retransmitido.

Este proceso de retransmisión en el interfaz *i* finaliza, o porque ya se ha retransmitido hasta el último paquete pendiente de confirmación, o porque se han recibido uno o más ACKs y *todos* sus hijos accesibles por el interfaz *i* le han asentido *todos* los paquetes que estaban pendientes de confirmación.

5.6.4 Filtrado de retransmisiones

Estas funciones de filtrado se realizan en los BRs y de forma genérica consisten en filtrar determinados paquetes de datos cuyo envío ha sido originado por una retransmisión. Dado que los procesos de retransmisión pueden originarse por el vencimiento de un temporizador o por una solicitud explícita, existen dos tipos de filtrado de retransmisiones, y se aplicará uno u otro dependiendo del origen del proceso de retransmisión. En ambos casos, las funciones de filtrado están asociadas a interfaces en lugar de a hijos; la motivación de esta decisión de diseño es hacer que todos los paquetes de datos que fluyen desde la fuente a los miembros, se propaguen a través del árbol de distribución, y esto tiene como finalidad, detectar lo antes posible los cambios en el árbol de distribución que pueden afectar al árbol RMNP. Debe observarse que si el filtrado se hiciera por hijo las retransmisiones deberían enviarse como paquetes punto a punto desde el padre a los hijos.

5.6.4.1 Filtrado de retransmisiones por temporizador

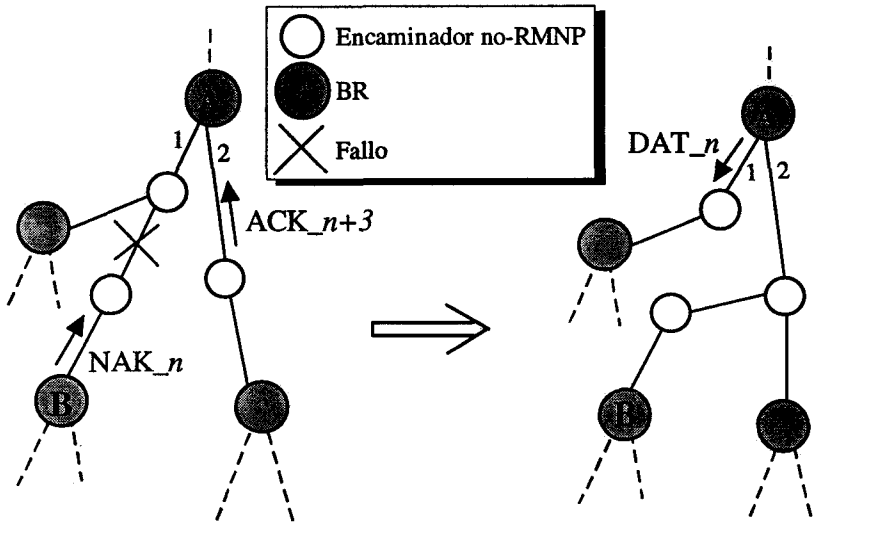
Un BR que está distribuyendo paquetes a sus hijos RMNP, ya sea en proceso básico (ver sección 5.4.1) o con NAKs pendientes (ver sección 5.6.3), al recibir un paquete previamente transmitido si estima que la razón de que haya vuelto a recibir dicho paquete ha sido un proceso de retransmisión por temporizador iniciado en un SR o en la fuente, únicamente distribuye dicho paquete por aquellos interfaces por los cuales están accesibles hijos que aún no le han confirmado dicho paquete. Cuando un BR recibe un paquete de datos, previamente recibido, cuya retransmisión no había sido solicitada explícitamente, asume que se trata de una retransmisión por temporizador.

5.6.4.2 Filtrado de retransmisiones por solicitud explícita

Un BR que está distribuyendo paquetes mientras tiene NAKs pendientes de satisfacer (ver sección 5.6.3), al recibir un paquete solicitado, únicamente lo envía por aquellos interfaces hijo por los qué están accesibles uno o más hijos que le habían solicitado previamente dicho paquete.

El realizar las funciones de filtrado de retransmisiones por temporizador y por solicitud explícita, ahorra recursos de red, pero puede originar determinadas situaciones de bloqueo analizadas a continuación. Supóngase que mientras un hijo está pendiente de recibir una retransmisión, hay un cambio en el árbol de distribución, que no provoca un cambio en el árbol RMNP (como se verá posteriormente esto significa que no se pondrá en marcha

un reinicio), pero causa que dicho hijo que antes estaba accesible por un determinado interfaz ahora esté accesible por otro distinto, este cambio de interfaz puede originar que el padre RMNP (BR) al aplicar el filtrado envíe los paquetes retransmitidos únicamente por el interfaz antiguo con lo cual dicho hijo no recibirá los paquetes que está esperando, y consecuentemente tampoco los confirmará, provocando una situación de bloqueo. En el ejemplo de la Figura 5.7, *B* estaba accesible por el interfaz hijo *1* y tras el cambio en el árbol de distribución queda accesible por el interfaz hijo *2*; y como puede observarse, al realizar *A* las funciones de filtrado de retransmisiones por solicitud explícita primero y de filtrado de retransmisiones por temporizador después, *B* no recibirá los paquetes solicitados. La forma de impedir tales situaciones es evitando que los BRs apliquen *siempre* el filtrado de retransmisiones por temporizador; en concreto, si el número de veces que un BR ha retransmitido un mismo paquete por un determinado interfaz ya ha superado un cierto umbral (MAX_FIL[15]), irán alternándose retransmisiones con filtrado y sin filtrado (se retransmite por todos los interfaces). En la sección 5.8 se expone como el hijo tras detectar el cambio en el interfaz, actualiza su información de estado y se lo notifica a su padre, el cual al recibir esta notificación también actualiza su información de estado y en concreto el interfaz asociado a dicho hijo.



a) Configuración del árbol de distribución antes del fallo.

b) Configuración del árbol de distribución después del fallo.

Figura 5.7: Un hijo cambia de interfaz

[15] Parámetro de configuración.

Este mismo problema de bloqueo puede originarse en un sistema retransmisor (fuente o SR) debido a que las retransmisiones por temporizador están asociadas a interfaces y un hijo puede cambiar de interfaz en la mitad de una comunicación por una modificación en el árbol de distribución. Análogamente, esto se evita utilizando el parámetro MAX_FIL, de modo que si el número de veces que un sistema retransmisor ha retransmitido un mismo paquete por un determinado interfaz *i* ya ha superado MAX_FIL, al expirar el temporizador de retransmisión asociado al mencionado paquete en dicho interfaz, los paquetes pendientes de retransmisión no se envían únicamente por el interfaz *i* sino por todos los interfaces hijo.

5.6.5 Distribución de paquetes con NAKs pendientes

En esta sección se analiza el comportamiento de los BRs cuando están distribuyendo paquetes y existen NAKs pendientes de satisfacer.

Dado un BR que está en un proceso de distribución básica de paquetes (ver sección 5.4.1), en el que por ejemplo el paquete esperado era el *n+m*, si recibe una solicitud de retransmisión desde cualquiera de sus hijos RMNP, por ejemplo recibe un NAK_*n*, detiene el proceso de distribución básica de paquetes y considera los paquetes en el rango [*n*, *n+m-1*] como *pendientes de retransmisión*. Cuando el encaminador dé por finalizado este proceso de recuperación, continuará con la distribución básica de paquetes a partir del paquete *n+m*.

Un BR, que tiene uno o más NAKs pendientes de satisfacer, al recibir un paquete de datos debe discernir (ver Apéndice B, página 238), utilizando la información de estado ya comentada, si éste es:

1. El paquete *esperado*. El siguiente paquete en secuencia de los pendientes de retransmisión.

2. Un paquete *fuera de secuencia*. Un paquete pendiente de retransmisión pero que se salta la secuencia.

3. Un paquete *duplicado no confirmado*. Un paquete en el que se cumplen las tres siguientes condiciones: primero, que ha sido transmitido a sus hijos previamente, segundo, que no ha sido confirmado a su padre RMNP, y tercero, que no está pendiente de ser retransmitido.

4. Un paquete *duplicado ya confirmado*. Un paquete en el que se cumplen las tres siguientes condiciones: primero, que ha sido transmitido a sus hijos previamente, segundo que ya ha sido confirmado a su padre RMNP, y tercero, que no está pendiente de ser retransmitido.

5. Un paquete *no transmitido previamente*. Un paquete que no ha sido enviado previamente a sus hijos.

Mientras existen NAKs pendientes de satisfacer, el comportamiento de los BRs variará según distintos sucesos:

- Cuando recibe un *paquete de datos* actúa de un modo u otro, dependiendo de:

  - Si es el paquete *esperado,* se lo reenvía únicamente a aquellos hijos que habían solicitado explícitamente dicho paquete, aplicando el *filtrado de retransmisiones por solicitud explícita*. Seguidamente, el BR se mantiene esperando el siguiente paquete en secuencia de los pendientes de retransmisión.

  - Si el paquete recibido está *fuera de secuencia,* actúa de forma análoga a lo visto en la sección 5.4.2. En esencia esto consiste en lo siguiente: almacena momentáneamente el paquete, activa el temporizador de fuera de secuencia y si éste expira, envía un NAK a su padre solicitándole la retransmisión a partir del paquete esperado.

  - Si es un paquete *duplicado no confirmado,* supone que este paquete ha sido originado por el vencimiento de un temporizador de retransmisión y aplica la función de *filtrado de retransmisiones por temporizador.*

  - Si es un paquete *duplicado ya confirmado*, lo descarta y reenvía el último ACK.

  - Si es un paquete *no transmitido previamente,* lo descarta considerando que éste continúa la secuencia en la distribución normal interrumpida.

- Si *recibe un NAK_k* desde un hijo RMNP, nótese que previamente se puede o no haber recibido otro NAK desde ese hijo concreto, actualiza la información de estado asociada a dicho hijo y, según lo visto en la sección de *filtrado de NAKs,* decide si debe reenviar dicho NAK a su padre.

- Si *recibe un ACK_n*, actualiza la información de estado asociada al hijo que le envió dicho asentimiento y si procede realiza un proceso de agregación de ACKs.

Cuando no hay más paquetes pendientes de retransmisión el proceso finaliza y el BR prosigue con distribución básica de paquetes a partir del punto donde quedó interrumpida.

5.7 Inicio y finalización de la comunicación

El RMNP es un protocolo orientado a conexión, esto significa que existen una fase de establecimiento de la conexión, otra de transferencia de datos y una última de liberación de la conexión. Aunque conceptualmente existan tres fases diferenciadas, al inicio de la comunicación hay un solapamiento de las fases de establecimiento y transferencia de datos. Esto se debe al hecho de que RMNP para establecer una conexión, no usa paquetes de control específicos sino paquetes de datos con un determinado flag activado.

Previamente al establecimiento de la conexión, algunos o todos los miembros se habrán suscrito al grupo, usando procedimientos ajenos al RMNP (por ejemplo el IGMP). En la definición actual de RMNP, al seguir el modelo de grupo planteado por Deering, se permite que los miembros entren o salgan del grupo en cualquier momento. Aunque, como ya se ha comentado en la sección 4.3, los miembros no recibirán los datos enviados al grupo en los periodos que no pertenezcan a éste, esto es, antes de unirse o después de dejar el grupo.

Dependiendo del algoritmo de encaminamiento multipunto usado, el árbol de distribución se habrá construido previamente al establecimiento de la conexión, o se construirá bajo demanda en esta fase. Por ejemplo, si el algoritmo de encaminamiento multipunto usado es el CBT se habrá construido previamente, pero si es el MOSPF se construirá bajo demanda.

Con el fin de hacer más clara la exposición, antes de analizar como se establece una conexión RMNP, se detalla el proceso de construcción del árbol RMNP.

5.7.1 Construcción del árbol RMNP

El árbol RMNP se construye en la fase de establecimiento de la conexión, a raíz de los primeros paquetes enviados por la fuente. Todos los paquetes enviados por la fuente, in-

corporan en la cabecera dos campos que se utilizan para la construcción y el mantenimiento del árbol RMNP. Estos son: el campo *último sistema RMNP visitado* (de aquí en adelante campo LRV), que como su nombre indica guarda la identidad del último sistema RMNP que ha visitado el paquete, y el campo *interfaz usado* (de aquí en adelante campo I), que guarda un identificador local del interfaz que empleó el último sistema RMNP visitado para enviar dicho paquete.

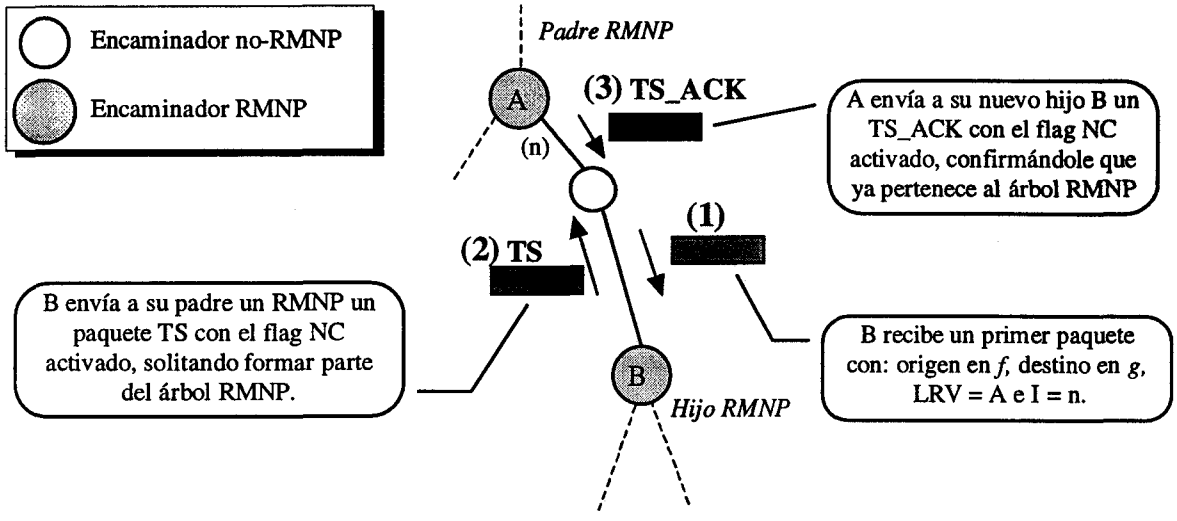


Figura 5.8: Construcción del árbol RMNP

Cuando un sistema RMNP recibe un primer paquete enviado por la fuente, distribuye este paquete a sus hijos en el árbol de distribución e inicia su proceso de unión al árbol RMNP (ver Figura 5.8). Analizando la información contenida en dicho paquete aprende, que su padre RMNP es aquel cuya identidad está anotada en el campo LRV, y que éste utiliza el interfaz identificado en el campo I para enviarle paquetes. Después de crear la correspondiente entrada en su TIB y anotar la información correspondiente, envía a su padre un paquete de *estado del árbol* (paquete TS) con el flag *nuevo hijo* (flag NC) activado, avisándole de su deseo de pertenecer al árbol RMNP. Este último paquete, que se manda como punto a punto al padre RMNP, incluye el identificador local del interfaz usado por dicho padre para mandarle paquetes. Nótese que el padre no tiene otra forma de conocer a través de cual de sus interfaces, de los pertenecientes al árbol de distribución, está accesible dicho hijo, ya que éste no tiene porque coincidir con el interfaz por el cual él mismo recibe los paquetes enviados como punto a punto por dicho hijo. El nuevo hijo, tras mandar el paquete TS, activa un temporizador llamado *esperando contestación de mi padre*. Si este temporizador expira volverá a enviar el paquete TS.

Un sistema RMNP al recibir un paquete TS con el flag NC activado, aprende sobre la existencia de un nuevo hijo y hace lo siguiente: (a) anota la información correspondiente en la TIB, identidad del hijo e interfaz asociado, (b) le envía un paquete TS_ACK (con el flag NC activado) informándole de que ya pertenece al árbol RMNP. Este paquete TS_ACK sirve para confirmar la recepción del paquete TS, y para que el padre informe a su nuevo hijo de cierta información de estado que éste debe conocer. En concreto le informa de tres cosas: (1) cual es el identificador del último reinicio conocido (ver sección 5.8), esta información está contenida en el campo *Identificador de reinicio* de la cabecera común; (2) cual es el estado del último reinicio conocido, todavía está activo o ya ha finalizado, esta información está contenida en el campo RS de la cabecera común; (3) cual ha sido el último paquete que ha confirmado el padre, esta información está contenida en el campo *último confirmado* (campo LC), y permite al hijo saber a partir de cual paquete debe esperar. Como es lógico, cuando un sistema se incorpora al árbol en la fase de establecimiento de la conexión, lo normal será que el campo LC tenga el valor de cero.

Debe puntualizarse que un encaminador RMNP puede recibir un paquete TS desde un hijo cuando él mismo todavía no pertenece al árbol RMNP, por no haber recibido el correspondiente TS_ACK desde su propio padre. En esta situación, el encaminador retarda el envío del TS_ACK hacia su hijo, hasta que él mismo ha recibido el TS_ACK desde su padre. Este comportamiento es lógico ya que un encaminador no puede indicar a su hijo a partir de cual paquete debe esperar, si él mismo no conoce esta información.

Un sistema que tenía pendiente la recepción de un TS_ACK, al recibirlo, actualiza cierta información de estado (el número de paquete esperado y el último reinicio conocido) y detiene el temporizador de *esperando contestación de mi padre.*

Cuando un miembro, que se comporta como un sistema RMNP más, recibe uno o más paquetes desde una fuente y no desea recibir datos desde dicha fuente, no envía un paquete TS hacia su padre RMNP, con lo cual no formará parte del árbol RMNP en construcción. Esto puede interpretarse como un rechazo del miembro a aceptar la conexión RMNP, ya que si no pertenece al árbol RMNP, ningún sistema es consciente de su existencia y consiguientemente no será tenido en cuenta en ningún proceso.

Este proceso inicial de construcción del árbol RMNP finaliza, cuando cada uno de los miembros ha recibido el TS_ACK enviado por su padre correspondiente. En secciones siguientes se analizará el mantenimiento del árbol RMNP. Este mantenimiento es originado por la entrada o salida de miembros en la fase de transferencia de datos, o por cambios en la topología de la red que provocan cambios en el árbol de distribución.

5.7.2 Establecimiento de una conexión RMNP

La fase de establecimiento de una conexión es asimétrica, es la fuente quién decide establecer una conexión siendo responsabilidad de los miembros aceptarla o no.

En esta fase se realizan los siguientes procesos:

- ❒ Se construye el árbol RMNP.
- ❒ Si el proceso de elección de SRs es dinámico, cada encaminador RMNP decide si realizará o no el papel de SR. La elección de SRs se comenta en la sección 5.12.
- ❒ Se inicializa la información de estado precisa en cada sistema y se reservan recursos (por ejemplo memoria).
- ❒ En el caso de que RMNP esté utilizando el algoritmo dinámico de descubrimiento de la MTTU (por no proporcionar este servicio el nivel inferior), la fuente usa este algoritmo al final de la fase de establecimiento. El hecho de descubrir de forma dinámica un valor para la MTTU al final de la fase de establecimiento en lugar de hacerlo al principio, se debe a que inicialmente el árbol RMNP no está construido y en estas condiciones no es apropiado utilizar el algoritmo planteado en la sección 5.4.4, ya que el valor obtenido podría no ser válido. Los paquetes enviados por la fuente en el establecimiento, tendrán un tamaño igual o menor a la MTU de la subred a la que está conectada la fuente.

La fuente establece una conexión a petición de una entidad de nivel superior. Como ya se ha comentado, para establecer una conexión, RMNP utiliza paquetes de datos en lugar de paquetes de control específicos. La fuente solicita el establecimiento de la conexión activando el flag *solicito conexión* (flag CR), en todos los paquetes pertenecientes a la primera ventana de transmisión excepto en el último. Esto significa, que si por ejemplo la ventana de transmisión tiene inicialmente un tamaño de 20, la fuente activará el flag CR de los primeros 19 paquetes. Más adelante se analiza la conveniencia de esta opción de diseño. La fuente empieza a numerar los paquetes de datos a partir del uno, por consiguiente, el primer paquete de la primera ventana de transmisión tendrá un uno como número de secuencia.

En RMNP, el conocimiento sobre la existencia e identidad de los miembros está distribuido por el árbol RMNP. Este conocimiento no es previo al establecimiento de la conexión sino que se va adquiriendo cuando cada miembro, después de recibir un primer pa-

quete distribuido por la fuente (que no tiene por qué coincidir con el primero enviado por ésta), informa a su padre RMNP de su existencia, al mismo tiempo que le indica su deseo de pertenecer al árbol. Como es lógico, si ningún sistema conoce la existencia de un determinado miembro, nadie se hará responsable de que éste reciba una copia de cada uno de los paquetes. Dado esto, en RMNP puede darse la siguiente situación (ver Figura 5.9): en un momento dado al inicio de la comunicación, un encaminador no conoce la existencia de algunos de sus descendientes y como consecuencia, cuando sus descendientes conocidos le confirman un determinado paquete, éste confirma dicho paquete a su padre. Este proceso de confirmación puede irse propagando rápidamente por el árbol, causando que se descarten la(s) copia(s) almacenada(s) de dicho paquete y se desplace la ventana correspondiente en la fuente, cuando sin embargo, alguno o algunos de los miembros no han recibido aún copia de dicho paquete.

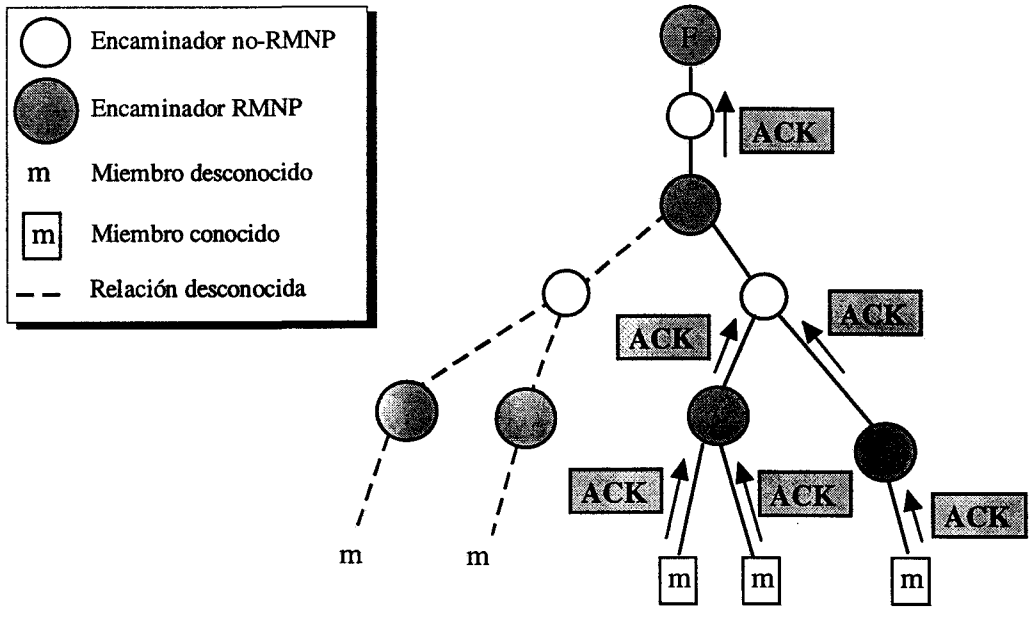


Figura 5.9: Propagación de los ACKs hacia la fuente

El primer objetivo de diseño de esta fase de establecimiento es minimizar la probabilidad de que un miembro, que se ha unido al grupo antes de que la fuente establezca la conexión, pierda alguno(s) de los primeros paquetes distribuidos por dicha fuente. Como se ha visto, esto puede deberse a que ningún sistema conoce inicialmente la existencia de este miembro y cuando es conocida, la fuente ya ha desplazado la ventana sin haberle tenido en consideración. Como se verá este objetivo de diseño se logra, a costa de introducir cierta sobrecarga de transmisión en esta fase de establecimiento.

Al seguir el modelo de Deering los miembros pueden incorporarse al grupo, bien previamente a la fase de establecimiento, o bien en la mitad de la fase de transferencia. Este hecho tiene dos consecuencias que deben ser consideradas: (1) inicialmente al unirse al grupo, un sistema puede no conocer en que fase de la comunicación se ha unido al árbol RMNP; en concreto, hasta que no recibe, o bien un paquete de datos con el flag CR activado (lo cual indica que se está estableciendo la conexión), o bien dos paquetes de datos distintos con el flag CR desactivado (lo cual indica que ha finalizado el establecimiento), no conoce en que fase está la comunicación; (2) inicialmente, un sistema no tendrá la información de estado correctamente actualizada; por ejemplo, no sabe con certeza cual debe ser el paquete esperado, esta información en concreto la conoce de forma precisa cuando su padre le envía un TS_ACK. En un principio, todos los sistemas suponen que se han unido al árbol en la fase de establecimiento y están esperando el paquete con número de secuencia uno.

Dado lo anterior, el segundo objetivo de diseño es evitar que la lógica de la fase de establecimiento sea excesivamente compleja, ya que una gran complejidad en la lógica puede conducir innecesariamente, a un protocolo ineficiente.

Vistos los objetivos de diseño, a continuación se expone y analiza, el comportamiento de los miembros y los encaminadores en la fase de establecimiento de la conexión. Primeramente se verá el comportamiento de los **miembros**:

- *Retardan la confirmación de los paquetes de datos recibidos.* Este comportamiento evita el problema, ya analizado, de que la fuente desplace demasiado deprisa la ventana, sin ser tenidos en consideración parte de los miembros. De acuerdo con esto, los miembros *retardan la confirmación de los paquetes recibidos correctamente, hasta haber recibido la primera ventana completa.* Debe notarse que los miembros no conocen el tamaño de la ventana en la fuente, sin embargo, el hecho de haber recibido la primera ventana completa será conocido al recibir en secuencia un primer paquete con el flag CR desactivado. Como es lógico, para que un miembro confirme paquetes debe pertenecer al árbol RMNP y esto implica que debe haber recibido un TS_ACK enviado por su padre; y en caso de no ser así, retrasará el envío de la confirmación hasta haber recibido dicho paquete.

  Puede darse el caso de que la entidad de nivel superior desee enviar pocos paquetes de datos, y en concreto, un número menor al tamaño de la primera ventana de transmisión. En esta situación, se recibirá del nivel superior una solicitud de liberación estando todavía en la fase de establecimiento. Ante este hecho, la fuente reenvía todos los paquetes pero esta vez, el último llevará el bit CR desactivado provo-

cando que los miembros inicien el proceso de confirmación. Como se verá al analizar el proceso de liberación de la conexión, una vez que la fuente haya recibido la confirmación a estos paquetes, atenderá la petición del nivel superior y enviará un paquete solicitando a los miembros la liberación de la conexión.

- *Sólo aceptan paquetes en secuencia y no generan NAKs.* El guardar temporalmente los paquetes recibidos fuera de secuencia y el generar NAKs en esta fase de establecimiento, complica mucho la lógica del protocolo sin obtenerse a cambio grandes beneficios.

  El hecho de no generarse NAKs implica, que en la fase de establecimiento, los paquetes perdidos sólo se recuperan por el vencimiento de temporizadores. El único efecto producido por esta decisión es un cierto retardo en el proceso de recuperación de paquetes.

Los **encaminadores** tienen el siguiente comportamiento durante el establecimiento de la conexión:

- *No hacen control intermedio de secuencia.* Esto implica que distribuyen a todos sus hijos en el árbol de distribución todos los paquetes recibidos. Este comportamiento tiene el siguiente efecto beneficioso: favorece que los miembros reciban lo antes posible un primer paquete (aunque éste no esté en secuencia), lo cual provocará que estos miembros soliciten antes pertenecer al árbol RMNP. Debe notarse que además de ser beneficioso, inicialmente y hasta que su padre RMNP no le envía un TS_ACK, un encaminador no conoce con certeza cual debe ser el paquete esperado, y por lo tanto, no puede realizar control intermedio de secuencia.

- *No generan NAKs.* Este comportamiento tiene dos justificaciones: (1) si los miembros no generan NAKs, carece de sentido que éstos sean generados por los sistemas intermedios; (2) si no se hace control intermedio de secuencia en los encaminadores, éstos carecen de la información necesaria para saber cuándo deben generar un NAK.

- *No hacen filtrado de retransmisiones.* Si no se generan NAKs no tiene sentido hacer filtrado de retransmisiones por solicitud explícita. Tampoco tiene sentido hacer filtrado de retransmisiones por temporizador dado que: (1) el proceso de establecimiento trata de favorecer el hecho de que *todos* los miembros reciban lo antes posible un primer paquete; (2) los miembros retardan el proceso de confirmación de paquetes hasta haber recibido la primera ventana completa.

- *El comportamiento de los BRs y los SRs es idéntico.* Esto significa que los SRs no realizan las funciones de almacenamiento y retransmisión. Este planteamiento tiene dos justificaciones: (1) permitir que ciertos sistemas intermedios (los SRs) almacenen los paquetes pendientes de confirmación en la fase de establecimiento, complica mucho la lógica del protocolo sin obtenerse a cambio grandes beneficios; (2) dado que en la fase de establecimiento la recuperación de paquetes se hace únicamente por el vencimiento de temporizadores y se retarda el proceso de confirmación, no tiene mucho sentido permitir que expiren los temporizadores asociados al mismo paquete, en la fuente y en los SRs.

Los miembros y los encaminadores RMNP detectan que ha finalizado la fase de establecimiento, cuando han recibido dos paquetes distintos con el flag CR desactivado. A partir de este momento, comienza su comportamiento normal, ya visto en las secciones anteriores.

Como se comentó en la sección 4.3, al analizar la problemática de proporcionar un servicio fiable cuando se sigue el modelo de Deering, uno de los planteamientos del RMNP es indicar al nivel superior la finalización de determinados periodos en los que es imposible garantizar una fiabilidad completa. De aquí, que cuando la fuente recibe confirmación a los paquetes pertenecientes a la primera ventana, indica al nivel superior el final de la fase de establecimiento de la conexión.

Existen otros protocolos que plantean esta misma solución de utilizar la fase de establecimiento de la conexión para empezar a mandar datos; por ejemplo, en el protocolo X.25 cuando se solicita la facilidad de selección rápida. Sin embargo, la motivación en ambos casos (X.25 y RMNP) es algo distinta. En X.25 esto se hace para reducir la excesiva sobrecarga introducida por las fases de establecimiento y liberación, con relación a la fase de transferencia de datos, cuando el volumen de datos a mandar es muy pequeño. Mientras que en RMNP, este planteamiento de utilizar paquetes de datos en lugar de paquetes de control permite: (1) minimizar la probabilidad de que un miembro no detecte la fase de establecimiento, ya que todos los paquetes de la primera ventana excepto el último llevan el flag CR activo, (2) hacer que los miembros conozcan el tamaño de la ventana en la fuente, simplemente al detectar un primer paquete en secuencia con el flag CR inactivo; y todo esto, introduciendo una sobrecarga mínima. Nótese que si se mandase un paquete de control que incluyera el tamaño de la ventana en la fuente, solicitando el establecimiento de la conexión, éste debería mandarse n veces para minimizar la probabilidad de que un miembro no lo recibiera, hecho que introduciría una sobrecarga innecesaria en la fase de establecimiento.

5.7.3 Entrada y salida de miembros en la fase de transferencia de datos

Primeramente se expone el proceso de mantenimiento del árbol RMNP, que es originado cada vez que un miembro entra o sale del grupo. Este proceso de mantenimiento es semejante, al proceso ya visto de construcción inicial del árbol RMNP, aunque con el fin de clarificar la exposición se analiza de forma separada. Posteriormente, se comentará el comportamiento de los sistemas cuando entran a formar parte del grupo, y no saben en que punto se haya la comunicación (establecimiento, transferencia o incluso liberación).

Dependiendo de la localización geográfica del miembro que entra o sale, con respecto al resto de miembros y a la fuente, pueden darse dos casos: o bien, que el cambio en el árbol RMNP afecte únicamente a dicho miembro, o bien, que afecte a éste y a uno o más encaminadores RMNP. Con el fin de clarificar la exposición, estos dos casos se analizan de forma separada aunque esto no implica que los procedimientos usados sean diferentes.

En primer lugar, se analiza la problemática planteada, cuando **el cambio en el árbol RMNP afecta únicamente al miembro que entra/sale del grupo**, para ello se tratan de forma separada la entrada y la salida de miembros.

❒ El proceso de *entrada de un miembro* será el siguiente:

- ♦ El miembro se suscribe al grupo usando procedimientos ajenos al RMNP, como por ejemplo el IGMP. En algunos casos esta nueva suscripción provoca que uno o más encaminadores no-RMNP, que no formaban parte del árbol de distribución, se incorporen a éste. Sin embargo, este hecho es transparente al nivel RMNP.

- ♦ Pasado un cierto tiempo, el nuevo miembro recibe un primer paquete distribuido por la fuente (paquete de datos ó MTTUD), analizando la cabecera de este paquete aprende la identidad de su padre RMNP y por cual interfaz está accesible, con esta información hace lo siguiente:

  - Crea una entrada en la TIB, en la cual anota la información relativa a dicho padre.

  - Envía a su padre RMNP un paquete TS, con el flag NC activado, avisándole de su deseo de pertenecer al grupo y por cual interfaz está accesible.

- Activa el temporizador de *esperando contestación de mi padre* mientras espera recibir el correspondiente TS_ACK (con el flag NC activado).

♦ El padre correspondiente al recibir el paquete TS hace lo siguiente: (1) actualiza la entrada asociada en la TIB, anotando la identidad de su nuevo hijo y el interfaz asociado, (2) envía un TS_ACK a su nuevo hijo. Como ya se ha comentado, el campo LC de dicho paquete permite al padre informar de cual ha sido el último paquete que ha confirmado, y por consiguiente a partir de cual paquete debe esperar el hijo. Como es lógico, ya que el miembro se está uniendo al árbol en la mitad de la fase de transferencia el primer paquete a esperar será generalmente un *n* distinto al paquete con número de secuencia uno.

❒ Proceso de *salida de un miembro*:

♦ Todo comienza cuando el miembro hace lo siguiente:

- Envía a su padre RMNP un paquete TS con el flag de *salida de un miembro* (flag ML) activado, solicitando salir del árbol.

- Activa el temporizador de *esperando contestación de mi padre* mientras espera recibir el correspondiente TS_ACK (con el flag ML activado).

- Cuando reciba el TS_ACK, borrará la entrada asociada en la TIB.

- Abandona el grupo usando un protocolo ajeno a RMNP, como por ejemplo el IGMP.

♦ Cuando el padre recibe el paquete TS actualiza la entrada asociada en la TIB, borrando la información relativa a dicho hijo, y le envía el correspondiente TS_ACK.

En segundo lugar, se analiza la problemática planteada cuando **el cambio en el árbol RMNP afecta al miembro que entra/sale del grupo y a uno o más encaminadores RMNP**, para ello se tratan de forma separada la incorporación y la salida de miembros.

❒ Proceso de *entrada de un miembro* a un grupo (ver Figura 5.10):

♦ El miembro (en el ejemplo de la Figura 5.10: m_{10}) se suscribe al grupo usando procedimientos ajenos al RMNP, como por ejemplo el IGMP. Esta nueva sus-

cripción provoca que uno o más encaminadores RMNP que no formaban parte del árbol de distribución se incorporen a éste (en el ejemplo de la Figura 5.10: E y F). Nótese que también puede ocurrir que uno o más encaminadores no-RMNP se incorporen al árbol de distribución (en el ejemplo de la Figura 5.10: 11), pero este último hecho es transparente al nivel RMNP.

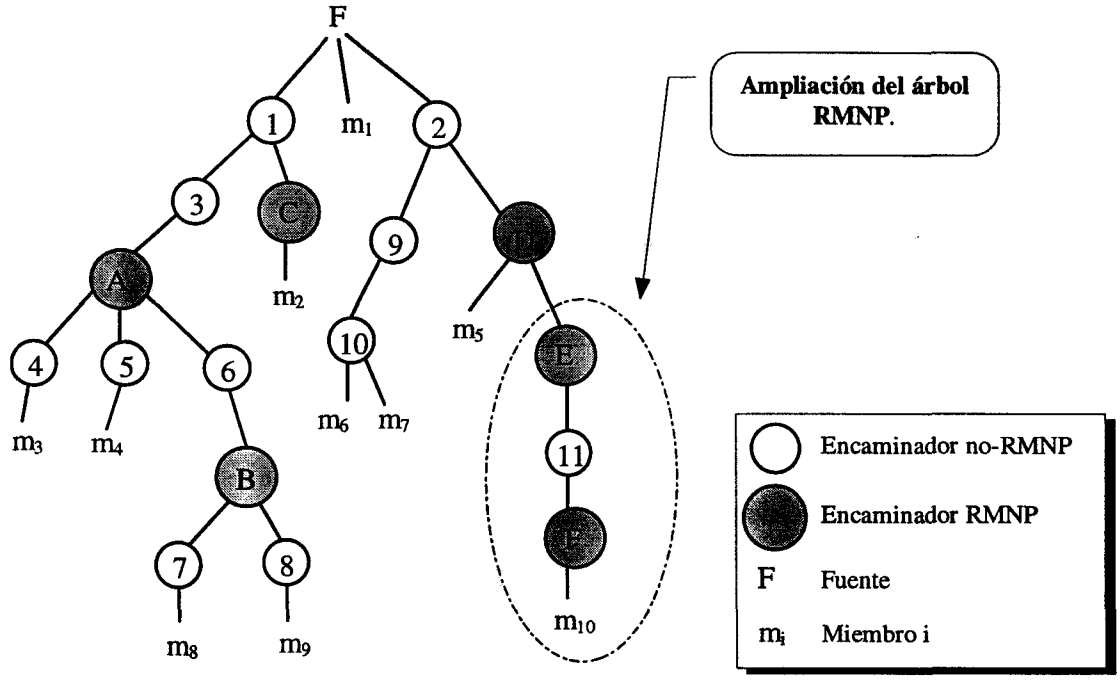


Figura 5.10: Incorporación de un miembro

- Pasado un cierto tiempo cada encaminador RMNP que se ha incorporado al árbol de distribución (en el ejemplo de la Figura 5.10: E y F), recibe un primer paquete distribuido por la fuente. Analizando la cabecera de este paquete aprende la identidad de su padre RMNP y por cual interfaz esta accesible, y hace lo siguiente:

  - Continúa con el proceso de propagación del paquete recibido por el árbol de distribución.

  - Crea una entrada en la TIB, en la cual anota la información relativa a su padre.

- Envía a su padre RMNP un paquete TS, con el flag NC activado, avisándole de su deseo de pertenecer al grupo e informándole de por cual interfaz está accesible.

- Activa el temporizador de *esperando contestación de mi padre* mientras espera recibir el correspondiente TS_ACK.

- Si recibe otros paquetes distribuidos por la fuente, también los propaga por el árbol de distribución.

- El padre correspondiente al recibir el paquete TS hace lo siguiente: (1) actualiza la entrada asociada en la TIB, anotando la identidad de su nuevo hijo y el interfaz asociado, (2) envía un TS_ACK con el flag NC activado a su nuevo hijo. Al igual que se ha visto en el proceso de construcción inicial del árbol, un encaminador retarda el envío de un TS_ACK a su hijo, hasta que él mismo no ha recibido un TS_ACK de su propio padre.

♦ Pasado un cierto tiempo, el nuevo miembro también recibe un primer paquete distribuido por la fuente y analizando la cabecera de este paquete aprende la identidad de su padre RMNP. El proceso realizado será semejante al comentado para un encaminador RMNP, excepto que el miembro no continúa con la propagación del paquete de datos por el árbol de distribución.

El proceso de ampliación del árbol RMNP finaliza en el momento en que el nuevo miembro recibe el TS_ACK. Como ya se ha comentado en el campo LC de dicho paquete permite al padre informar de cual ha sido el último paquete que ha confirmado, y por consiguiente a partir de cual paquete debe esperar el hijo.

❒ Proceso de *salida de un miembro* de un grupo (ver Figura 5.11):

♦ Todo comienza por iniciativa del miembro (en el ejemplo de la Figura 5.11: m_{10}) que hace lo siguiente:

- Envía a su padre RMNP un paquete TS con el flag ML activado, solicitando salir del árbol.

- Activa el temporizador de *esperando contestación de mi padre* mientras espera recibir el correspondiente TS_ACK.

- Cuando reciba el TS_ACK borrará la entrada asociada en la TIB.

- Abandona el grupo usando un protocolo ajeno al RMNP, como por ejemplo el IGMP. Este abandono origina que uno o más encaminadores RMNP que formaban parte del árbol de distribución salgan de éste (en el ejemplo de la Figura 5.11: E y F). Nótese que también puede ocurrir que uno o más encaminadores no-RMNP salgan del árbol de distribución (en el ejemplo de la Figura 5.11: 11); sin embargo, este último hecho es transparente al nivel RMNP.

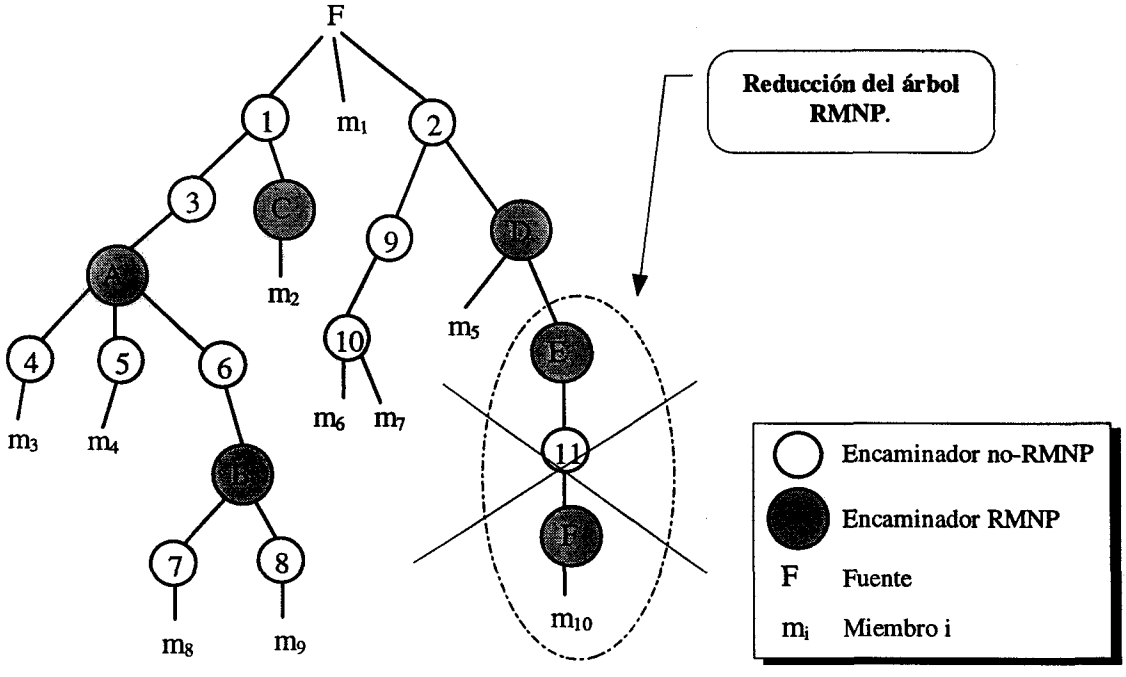


Figura 5.11: Salida de un miembro

- Cuando el padre (en el ejemplo de Figura 5.11: F) recibe el paquete TS hace lo siguiente:

  - Actualiza la entrada asociada en la TIB, borrando la información relativa a dicho hijo.

  - Envía a su hijo el TS_ACK correspondiente indicándole que ha recibido su solicitud de salir del árbol.

- Si no tiene más hijos en la entrada asociada en la TIB, envía a su propio padre RMNP un paquete TS solicitando salir del árbol (con el flag ML activado), y activa el temporizador de *esperando contestación de mi padre* mientras espera recibir el correspondiente TS_ACK.

♦ Este proceso se va repitiendo y finaliza cuando un encaminador comprueba que tiene otros hijos en la correspondiente entrada en la TIB (en el ejemplo de Figura 5.11: D), con lo cual, no solicita a su propio padre salir del árbol.

En el caso de que el miembro que ha solicitado salir del grupo fuera el último miembro de dicho grupo, todos los encaminadores RMNP, que están en el camino entre la fuente y dicho miembro, irán solicitando su salida del árbol RMNP. Finalmente, la fuente detectará que no tiene ningún hijo RMNP, y avisará al usuario del servicio de que la conexión ha sido abortada indicándole la causa.

El comportamiento inicial de los sistemas cuando entran a formar parte del árbol RMNP, una vez iniciada la comunicación, viene determinado por el hecho de que en un primer momento, éstos no son capaces de discriminar si se han unido al grupo, previamente a que finalizara la fase de establecimiento, o ha sido en fase de transferencia de datos. De aquí, que inicialmente asuman que está estableciéndose la comunicación y se comporten como se ha visto en la sección 5.7.2 (Establecimiento de una conexión RMNP). Una vez que estos nuevos sistemas ya pertenecen al árbol RMNP y han recibido dos paquetes con el flag CR desactivado, ya conocen que la comunicación está en la fase de transferencia y comienza el comportamiento propio de esta fase.

5.7.4 Liberación de una conexión RMNP

La liberación de una conexión RMNP puede ser total o parcial. Se denomina *liberación total* a un proceso de liberación iniciado por la fuente, éstos procesos tienen como objetivo dar por completada totalmente la conexión RMNP; y se denomina *liberación parcial,* a aquellos procesos de liberación iniciados por un encaminador que tienen como objetivo expulsar del árbol RMNP a uno o más miembros que están teniendo un comportamiento anómalo (por ejemplo, después de retransmitirles cierto número de veces un mismo paquete de datos, éstos no lo confirman). En primer lugar se analiza la liberación total para pasar a exponer la liberación parcial.

El principal propósito de una liberación total es liberar los recursos asociados a la conexión en los distintos sistemas (fuente, encaminadores y miembros). La liberación total de

una conexión puede ser debida a dos situaciones bien distintas, o bien, a que el usuario del servicio no tiene más datos que enviar y solicita liberar la conexión, liberación denominada *ordenada*, o bien, a una situación anómala que obliga a liberar la conexión, liberación denominada *aborto* de la conexión.

Cuando el nivel superior solicita una liberación ordenada de la conexión, la entidad RMNP en la fuente espera recibir el acuse de recibo de los paquetes que tiene pendientes de confirmación, antes de iniciar el correspondiente proceso de liberación. Una vez que le han sido confirmados todos estos paquetes, envía un paquete LIB con el flag de *liberación ordenada* (flag OR) activo. Seguidamente activa el temporizador de retransmisión, y se mantiene a la espera de recibir el asentimiento correspondiente (llamado LIB_ACK). Si expira el temporizador de retransmisión sin haber recibido un paquete LIB_ACK, retransmite el paquete LIB. Si dicho temporizador expira MAX_LIB[16] veces, sin haberse recibido el correspondiente LIB_ACK, la fuente libera los recursos locales asociados a la conexión e indica lo sucedido al nivel superior. En el caso de recibir el mencionado LIB_ACK, informa al nivel superior de que la liberación ordenada finalizó con éxito.

Los encaminadores RMNP realizan un proceso de agregación con los paquetes LIB_ACKs similar, aunque más sencillo, al realizado con los ACKs. Cuando un encaminador ha recibido un paquete LIB_ACK desde cada uno de sus hijos, envía un LIB_ACK hacia su padre.

En el caso de que la fuente, tras detectar una situación anómala, ponga en marcha un proceso de aborto de la conexión, no espera a recibir confirmación a los paquetes que tiene pendientes de confirmación y envía un paquete LIB con el flag de *aborto* (flag AB) activado. La fuente libera los recursos asociados a la conexión e indica al nivel superior la finalización de la conexión, así como la causa.

Los procesos de liberación parcial son originados en los encaminadores y, como ya se ha dicho, tienen como objetivo expulsar del grupo a uno o más hijos que están teniendo un comportamiento anómalo. En realidad una liberación parcial es un proceso de aborto parcial pues el encaminador no espera a que se confirmen los paquetes pendientes. Un encaminador que pone en marcha un proceso de liberación parcial hace lo siguiente: (1) envía al hijo o hijos correspondientes un paquete LIB con el flag AB activado; (2) borra a dichos hijos de la TIB; (3) informa a la fuente de la liberación parcial enviando un paquete LIB con el flag de *información* (flag INF) activado, y pone en marcha el temporizador de *espe-*

[16] Parámetro de configuración.

rando contestación de la fuente mientras espera recibir el correspondiente LIB_ACK enviado por la fuente (que también llevará el flag INF activo).

Debido a distintas situaciones anómalas, puede darse el caso de que un sistema no reciba el paquete LIB y como consecuencia no libere los recursos asociados a una conexión. Esta problemática se resuelve, obligando a la fuente a no estar un periodo excesivamente largo sin distribuir ningún paquete por el árbol y, usando un temporizador en los encaminadores y los miembros. Por un lado, para garantizar que la fuente no está un periodo excesivamente largo sin distribuir ningún paquete por el árbol, la fuente dispone de un temporizador llamado de *estado de la fuente.* En el caso de que la fuente no tenga datos para enviar al grupo durante un periodo muy largo de tiempo y expire dicho temporizador, ésta enviará N_{SS}[17] paquetes de datos vacíos. Por otro lado, en los encaminadores RMNP y en los miembros se mantiene un temporizador denominado *temporizador de liberación;* cada vez que uno de estos sistemas recibe un paquete enviado por la fuente (paquete de datos lleno o vacío, o MTTUD), actualiza el correspondiente temporizador de liberación, asignándole el valor máximo. Si expira el temporizador de liberación asociado a una conexión, el sistema correspondiente asume que se ha producido una liberación de dicha conexión y libera los recursos locales asociados. En la sección 5.10 se analiza el problema del ajuste de los distintos temporizadores, pero se adelanta que para los temporizadores de estado de la fuente y de liberación, se recomienda un valor relativamente alto. El motivo de enviar varios paquetes de datos vacíos seguidos, es para minimizar la probabilidad de que un encaminador o un miembro asuma erróneamente que se ha liberado una conexión.

Aunque el principal motivo que justifica la existencia del temporizador de liberación y el envío de los paquetes de datos vacíos, es el hecho de que un sistema puede no conocer que se ha liberado una conexión al no recibir el correspondiente paquete LIB, existe otra motivación adicional. Un sistema RMNP, debido a una partición en la interred o a un cambio en la topología, puede dejar de formar parte del árbol RMNP y no detectarlo, esto tiene como consecuencia, que el sistema no liberará los recursos asociados a la conexión. El uso del temporizador de liberación y el envío de los paquetes vacíos también resuelve esta irregularidad.

[17] Parámetro de configuración.

5.8 Reinicio

Se pone en marcha un proceso de reinicio, cuando hay un cambio en el árbol de distribución que origina una modificación del árbol RMNP, excluyendo las modificaciones de éste provocadas por la entrada o salida de miembros. Una modificación en el árbol RMNP significa que hay uno o más cambios en las relaciones padre-hijo RMNP. Más en concreto, un cambio en una relación RMNP significa que un hijo que tenía un determinado padre RMNP, a partir de un cierto momento cambia de padre. En la mayoría de los casos, los cambios en el árbol de distribución son provocados por cambios en la topología, aunque en determinadas circunstancias pueden deberse a la aparición de rutas punto a punto más óptimas.

En las circunstancias expuestas, la información de estado RMNP mantenida en algún o algunos nodos puede ser inconsistente, y de aquí surge la necesidad de poner en marcha un procedimiento de reinicio que permita reconstruir el árbol RMNP, y actualizar correctamente la información de estado. A continuación se clarifica la afirmación anterior utilizando un ejemplo concreto.

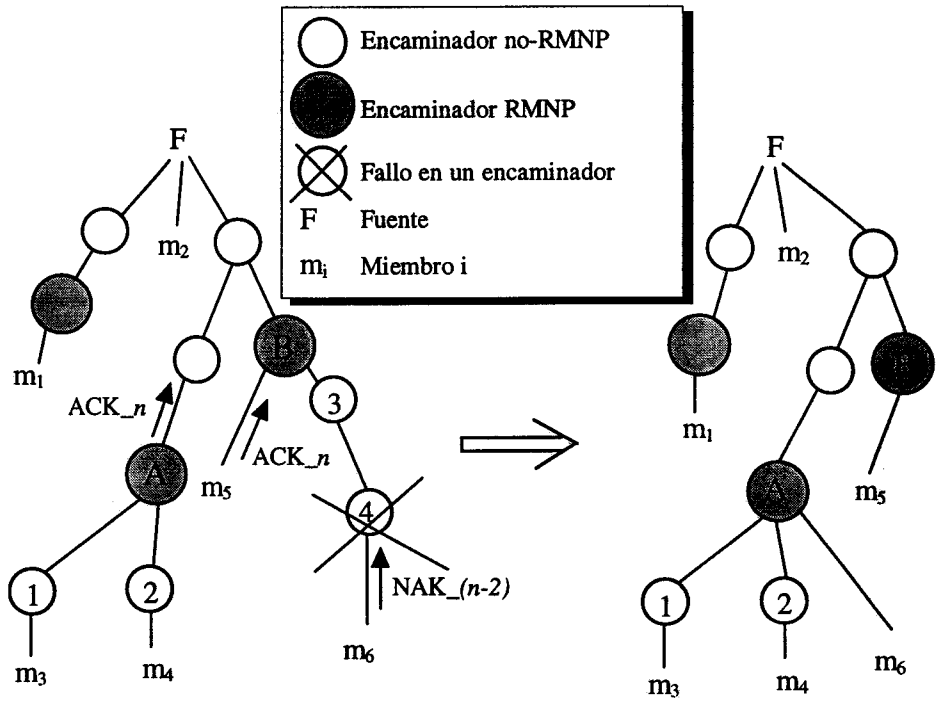


Figura 5.12: Reconfiguración del árbol de distribución

Dado el árbol de distribución de la Figura 5.12 la situación inicial era la siguiente: el nodo *A* tenía dos hijos RMNP (m_3 y m_4) y el nodo *B* tenía otros dos hijos (m_5 y m_6). El nodo *A* había confirmado a su padre RMNP (la fuente) hasta el paquete *n*, lo mismo había hecho m_5 con respecto a su propio padre; asimismo, m_6 había enviado a su padre un NAK_(*n-2*) solicitando la retransmisión a partir del paquete *n-2*. Tras un fallo en el encaminador *4*, hay una reconfiguración del árbol de distribución, tras la cual el nodo *B* tiene un único hijo (m_5) y el nodo *A* tiene tres hijos (m_3, m_4 y m_6). En esta situación ocurrirá lo siguiente:

- Si en la fuente expira el temporizador asociado al paquete *n-2*, ésta reenviará a partir del paquete *n-2* a *A* y *B* (nótese que ambos sistemas comparten el mismo interfaz en la fuente). Al recibir dichos paquetes, *A* clasificará el *n-2*, *n-1* y *n* como *duplicados ya confirmados*, y no se los retransmitirá a sus descendientes. Dada esta situación, m_6 no recibirá los paquetes que le faltan.

- Por otro lado, a pesar de que m_5 ya había confirmado hasta el paquete *n*, *B* no reenvía a su propio padre dicha confirmación por estar esperando la correspondiente confirmación proveniente de m_6.

Del ejemplo expuesto, puede deducirse que en la fase de reinicio será necesario: (1) reconstruir el árbol RMNP, y esto significa que los distintos sistemas deben comprobar que sus relaciones padre-hijo anteriores son correctas y aprender las posibles nuevas relaciones aparecidas; (2) actualizar la información de estado relativa a cada hijo (por ejemplo, cual es el último paquete que éste ha confirmado).

Cada procedimiento de reinicio está identificado por un número de secuencia, de modo que reinicios sucesivos tendrán identificadores consecutivos. Por otro lado, cada sistema RMNP recuerda el identificador del último reinicio conocido, y esto permite que cuando un sistema detecta un cambio en el árbol RMNP, éste indique a la fuente, cual cree que debe ser el identificador del procedimiento de reinicio a comenzar.

Los cambios en el árbol RMNP son detectados por distintos sistemas RMNP (encaminadores o miembros) cuando reciben un paquete distribuido por la fuente (DAT o MTTUD), conteniendo en el campo LRV la identidad de un nodo RMNP distinta a la de su actual padre. En esta situación el sistema que detectó la anomalía hace lo siguiente:

- Envía a la fuente, como un mensaje punto a punto, un paquete de *estado del árbol* (paquete TS) con el flag de *reinicio necesitado* (flag RN) activo. Este paquete in-

cluye cual debería ser el identificador del procedimiento de reinicio a comenzar, a juicio del sistema que detectó el cambio.

- ❒ Activa el temporizador de *esperando contestación de la fuente* mientras espera recibir un paquete de datos (vacío o lleno), que le indique que la fuente ha puesto en marcha el proceso de reinicio solicitado. En el caso de que expire este temporizador, reenviará el anterior paquete TS hacia la fuente.

Es obvio que un mismo cambio en el árbol de distribución puede ser detectado, casi simultáneamente, por varios sistemas, sin embargo es altamente improbable que éstos produzcan un problema de implosión en la fuente al mandar los paquetes TS, pues el número de dichos sistemas no será muy elevado.

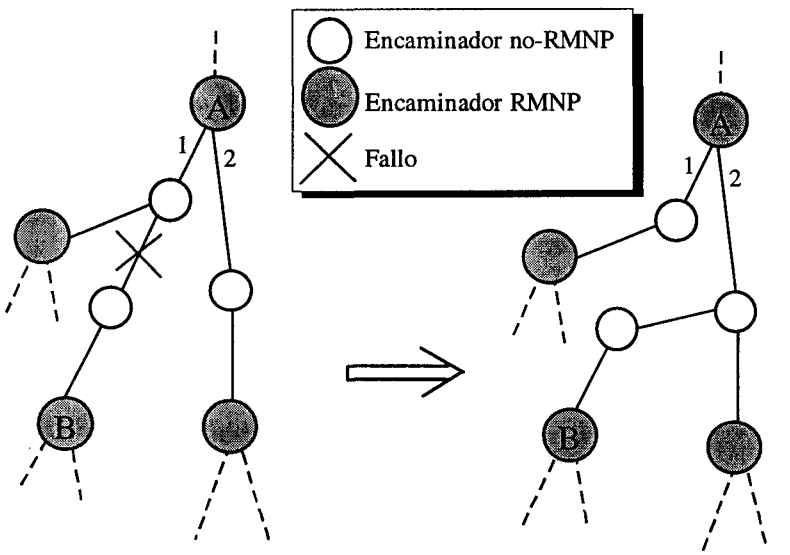
A continuación se exponen dos aspectos relacionados con el procedimiento de reinicio; primero se analiza por qué determinados cambios en el árbol RMNP, y en concreto los originados por la entrada o salida de miembros, no deben y no lo hacen, originar un procedimiento de reinicio; posteriormente, se razona y profundiza en el hecho de que no todos los cambios en el árbol de distribución provocan cambios en el árbol RMNP.

En el caso de la entrada o salida de un miembro, es obvio que habrá cambios en el árbol RMNP, en concreto aparecerán o desaparecerán ciertas relaciones padre-hijo (la TIB se modificará con los procedimientos vistos en la sección 5.7.3), sin embargo, debe resaltarse que el hecho de la entrada o salida de un miembro, en ningún caso causa que la información de estado quede inconsistente, y de aquí que no sea preciso un reinicio. Por otro lado, las modificaciones en el árbol RMNP originadas por la entrada o salida de miembros son perfectamente diferenciables de las causadas por cambios en la topología; nótese que cuando una modificación es debida a la entrada o salida de un miembro, no habrá ningún sistema que detecte que antes tenía un padre RMNP y ahora tiene otro distinto, y como consecuencia ningún sistema solicitará a la fuente un reinicio. Por el contrario si el cambio en el árbol RMNP se debe a un cambio en la topología, como consecuencia de la reconstrucción del árbol de distribución, al menos uno de los sistemas detectará que ha cambiado de padre, y avisará a la fuente. Debe comentarse que en el caso de que un cambio en el árbol RMNP originado por la entrada o salida de un miembro, provocara una fase de reinicio, el funcionamiento del RMNP con grupos dinámicos no sería adecuado.

Es obvio que no todos los cambios en el árbol de distribución provocan cambios en el árbol RMNP. De hecho, es posible que hayan modificaciones en el árbol de distribución que sean totalmente transparentes al nivel RMNP, y esto significa que ningún sistema RMNP detectará el cambio, pues éste no ha provocado la modificación de ninguna de las

relaciones padre-hijo RMNP existentes y tampoco ha causado la variación de ninguna información de estado. Existe una posible modificación en el árbol de distribución que merece especial atención y análisis, ésta es la siguiente: un cambio en el árbol de distribución que no origina ningún cambio en ninguna relación padre-hijo RMNP, pero que provoca que un hijo que antes estaba accesible para el padre por un determinado interfaz, a partir del cambio esté accesible por otro interfaz distinto (ver Figura 5.13). Por los motivos vistos al analizar el proceso de construcción del árbol, esta información no puede ser actualizada por el padre hasta que no se lo indique el hijo correspondiente.

El proceso seguido para actualizar la mencionada información será el siguiente: cuando el hijo correspondiente reciba un paquete distribuido por la fuente (DAT o MTTUD), conteniendo en el campo LRV la identidad de su padre, pero con un identificador de interfaz distinto al que él tenía almacenado en su TIB, envía a su padre RMNP un paquete TS, con el flag *nuevo interfaz* (flag NI) activado, indicándole cual es ahora el nuevo interfaz. El hijo activa el temporizador de *esperando contestación de mi padre* y se mantiene esperando a recibir un TS_ACK con el flag NI activado. El padre al recibir el mencionado paquete TS, actualiza la información relativa al interfaz y contesta a su hijo con un TS_ACK.



a) Configuración del árbol de distribución antes del fallo.

b) Configuración del árbol de distribución después del fallo.

Figura 5.13: Cambio en el árbol de distribución

En la situación planteada no es necesario un reinicio, pues salvo el identificador de interfaz, la información de estado relativa al hijo es correcta. El hecho de que el identificador de interfaz no esté correctamente actualizado en el padre, únicamente puede provocar

problemas transitorios, y esto sólo en el caso, de que cuando ocurra este cambio en el árbol, el padre esté aplicando la función de filtrado de retransmisiones. Esto es debido a que el filtrado de retransmisiones se hace por interfaz, y la información sobre la identidad de los hijos asociados a cada interfaz no es correcta. A continuación se analiza esta situación errónea transitoria con el ejemplo de la Figura 5.13. Supóngase que antes del cambio en el árbol de distribución, *B* hubiera solicitado la retransmisión de un determinado paquete. En esta situación podría ocurrir que al llegar dicho paquete a *A*, éste, al realizar el filtrado de retransmisiones y tener en su información de estado que "*B* está asociado al interfaz *1*", mandase el mencionado paquete por el interfaz *1* pero no por el interfaz *2*. Como es obvio, a consecuencia de este filtrado el paquete solicitado no le llegaría a *B*. Sin embargo, esto tan sólo será una situación transitoria pues, como se vio al analizar el problema de los filtrados de retransmisiones, después de un cierto tiempo (al aumentar el número de retransmisiones) *A* no aplicará filtrado y distribuirá el paquete por todos los interfaces. La recepción del mencionado paquete permitirá que *B* pueda detectar el cambio en el interfaz y comunicárselo a su padre (*A*). De todos modos, está situación será poco frecuente pues tiene que darse la coincidencia de que un hijo cambie de interfaz mientras su padre está realizando filtrado de retransmisiones, y más en concreto, el padre debe estar aplicando el filtrado al nuevo interfaz del hijo.

Una vez analizado por qué y cuándo es necesario poner en marcha un procedimiento de reinicio, y cómo son detectados los cambios en el árbol RMNP, se pasará a exponer el comportamiento de los distintos sistemas cuando está activo un procedimiento de reinicio. Antes se deben recordar ciertos aspectos que han sido claves en el diseño del procedimiento de reinicio:

1. *Es preciso reconstruir el árbol RMNP,* y con este fin, cada sistema intenta confirmar que la información que guarda en su TIB es la correcta.

2. *Es necesario inicializar y actualizar la información de estado* (distinta a la TIB) relativa al padre y a los distintos hijos, ya que ésta podría ser inconsistente.

3. *Se debe minimizar la probabilidad de que un hijo quede fuera del árbol RMNP reconstruido.* Debido a que en esta fase se reconstruye el árbol RMNP, y por las mismas razones justificadas en la sección de establecimiento de la conexión, el diseño del procedimiento de reinicio trata de minimizar la probabilidad de que un miembro, al no recibir ningún paquete de los distribuidos en el reinicio, no detecte que se está reconstruyendo el árbol y quede fuera de éste. De hecho, el comportamiento de los encaminadores y los miembros es muy similar al visto para el procedimiento de establecimiento.

Los paquetes RMNP incluyen un campo denominado *identificador de reinicio* (campo RI) y un flag llamado *estado del reinicio* (flag RS). Si RS está a uno, significa que el reinicio identificado en el campo RI está activo mientras que si está a cero significa que dicho reinicio finalizó. Estos campos permiten a los sistemas RMNP detectar, entre otras cosas, el inicio y la finalización de una fase de reinicio.

Un procedimiento de reinicio es puesto en marcha por la fuente a petición de uno o más sistemas RMNP que han detectado un cambio en el árbol. En concreto, la **fuente** al recibir un paquete TS con el flag de *reinicio necesitado* (flag RN) activado, tiene el siguiente comportamiento:

- ❒ Analiza el valor del campo RI. Si éste es igual al identificador del último reinicio conocido más uno, considera válida la solicitud de reinicio y como consecuencia pone en marcha un procedimiento de reinicio, que tiene como identificador el valor del campo RI del paquete TS recibido; en caso contrario, no da por válida la solicitud de reinicio y descarta el paquete. Gracias a este mecanismo si recibe varios paquetes TS, que están avisando del mismo cambio en el árbol RMNP, únicamente considera el primero mientras que el resto serán descartados.

- ❒ Si la fuente tenía en marcha un proceso de descubrimiento de MTTU, éste es detenido y, en el caso de ser necesario, se realizará después de finalizar el reinicio. Esta medida es necesaria ya que, al haber cambiado alguna o algunas relaciones padre-hijo y no estar actualizada correctamente la información en las TIBs de los distintos sistemas, el valor calculado para la MTTU podría no ser correcto.

- ❒ Inicializa el tamaño de las ventanas de transmisión asociadas a los distintos interfaces hijo a W_G, parámetro de configuración que indica cual es el tamaño de la ventana general.

- ❒ Si cuando la fuente decide poner en marcha el procedimiento de reinicio no tiene paquetes pendientes de confirmación, inicia un procedimiento denominado *reinicio simple*. Debe observarse que al no existir paquetes pendientes de confirmación toda la información de estado distinta a la TIB será consistente. El hecho de que la fuente ponga en marcha un reinicio simple significa que distribuye a los miembros, *N* paquetes vacíos, siendo *N* el tamaño de la ventana general. Estos paquetes tienen las siguientes particularidades: (1) en el campo RI (identificador de reinicio) incluyen el identificador del procedimiento de reinicio puesto en marcha; (2) llevan el flag RS (estado del reinicio) activo; (3) todos estos paquetes excepto el último lle-

van el flag CR activo; esto permite indicar a los miembros, en que momento deben confirmar cual es el último paquete recibido en secuencia.

- Si cuando la fuente recibe la indicación de que ha cambiado el árbol RMNP, ésta tiene paquetes pendientes de confirmación pondrá en marcha un verdadero procedimiento de *reinicio.* El hecho de que ponga en marcha un reinicio significa que reenvía a los miembros todos los paquetes que tiene pendientes de confirmación por algún interfaz (ventana general). Estos paquetes tienen las siguientes particularidades: (1) en el campo RI (identificador de reinicio) incluyen el identificador del procedimiento de reinicio puesto en marcha; (2) llevan el flag RS (estado del reinicio) activo; (3) todos estos paquetes excepto el último llevan el flag CR activo; esta última acción permitirá a los miembros detectar que la fuente ha terminado de reenviar todos los paquetes pendientes de confirmación. Debe notarse que cuando la fuente pone en marcha un procedimiento de reinicio, interrumpe temporalmente el envío de nuevos paquetes, hasta no dar por finalizado dicho procedimiento.

Los encaminadores y los miembros detectan que la fuente ha puesto en marcha un procedimiento de reinicio al recibir un paquete de datos (con datos o sin datos), con el campo RI distinto de cero y el flag RS activado; ante este hecho, actualizan su información de estado sobre el último reinicio conocido y dan por comenzado el procedimiento de reinicio. En general el comportamiento de los encaminadores y los miembros mientras está activo un proceso de reinicio es el siguiente:

- Todos los paquetes que generan, incluyen en el campo RI el identificador del procedimiento de reinicio activo.

- Sólo aceptan paquetes en secuencia y no generan NAKs (al igual que en la fase de establecimiento). Consecuentemente, la recuperación de los paquetes perdidos se hará por el vencimiento de temporizadores en la fuente.

- Los paquetes recibidos, que identifican en el campo RI a un reinicio anterior al activo o tienen este campo igual a cero, son descartados.

- Si reciben un paquete que identifica en el campo RI a un reinicio posterior al activo, hecho que significa que se ha producido un cambio en el árbol de distribución antes de que finalizara el reinicio en curso, actualizan su información de estado sobre el último reinicio conocido y dan por comenzado un nuevo procedimiento de reinicio.

Una vez vistas las generalidades, se expondrán los detalles específicos en el comportamiento de los encaminadores y los miembros mientras existe un procedimiento de reinicio activo.

El comportamiento específico de los **miembros**, al detectar que se ha puesto en marcha un procedimiento de reinicio, será el siguiente:

- Envían a su padre un paquete TS (estado del árbol), con el flag TC (comprobación del árbol) activado, indicándole que actualice su TIB. Seguidamente, activan el temporizador de *esperando contestación de mi padre,* y se mantienen a la espera de recibir el correspondiente TS_ACK (con el flag TC activo) confirmando que el padre actualizó su TIB.

- Retardan la confirmación de los paquetes recibidos correctamente, al igual que se hacía en la fase de establecimiento y además, con la misma motivación. En concreto, esta confirmación es retardada hasta recibir un primer paquete de datos con el flag CR desactivado. Como es obvio, en el caso de un reinicio simple (se han reenviado paquetes vacíos), los miembros volverán a confirmar el último paquete recibido en secuencia antes de iniciarse el procedimiento de reinicio. En el caso de recibir un primer paquete con el flag CR desactivado y aún no haber tenido contestación de su padre al proceso de comprobación de árbol (no haber recibido el TS_ACK), esperan a recibir ésta contestación antes de empezar a confirmar.

El comportamiento específico de un **encaminador**, al detectar que se ha puesto en marcha un procedimiento de reinicio, será el siguiente:

- Marca la información de la TIB relativa a sus hijos y a su padre, como pendiente de comprobar. En el caso de un SR también libera los buffers.

- Si el procedimiento en curso es un verdadero reinicio (ha sido detectado por una indicación contenida en un paquete de datos lleno), borra la información de estado (distinta a la TIB) relativa a sus hijos y a su padre. Como se ha visto esta información podría ser inconsistente y será necesario actualizarla durante el procedimiento de reinicio en curso.

- Envía a su padre un paquete TS (estado del árbol), con el flag TC (comprobación del árbol) activado, indicándole que actualice su TIB. Seguidamente, activa el temporizador de *esperando contestación de mi padre,* y se mantiene a la espera de recibir el correspondiente TS_ACK (con el flag TC activo) confirmando que el padre

actualizó su TIB. Al recibir dicho TS_ACK marca como comprobada la información de la TIB relativa a su padre.

- Espera a recibir de cada hijo un paquete TS con el flag TC activo, al ir recibiendo cada uno de estos paquetes, comprueba si la información en la TIB es incorrecta en cuyo caso la actualiza, y la marca como comprobada. Una vez hecho esto envía al hijo correspondiente la confirmación (TS_ACK).

- Al igual que en la fase de establecimiento y con la misma motivación, no hace control intermedio de secuencia ni filtrado de retransmisiones.

- Los SRs se comportan de forma idéntica a los BRs, de modo que no almacena los paquetes pendientes de confirmación ni ponen en marcha procesos de retransmisión.

- Cierto tiempo después de ponerse en marcha el procedimiento de reinicio, el encaminador empezará a recibir paquetes ACK de sus hijos. Al realizar el proceso de agregación de estos ACKs, el encaminador no tendrá en cuenta a aquellos hijos cuya información en la TIB aún no ha sido comprobada (no habrá sido comprobada al no haberse recibido el correspondiente TS). Estos ACKs, que son enviados al final del reinicio desde los miembros hacia la fuente, tienen como finalidad actualizar en los encaminadores la información de estado (distinta a la TIB) relativa a cada hijo y al padre.

La fuente da por finalizado el proceso de reinicio cuando recibe un primer paquete de confirmación, obviamente identificando en el campo RI al proceso de reinicio activo. Los siguientes W_G paquetes de datos enviados por la fuente, siendo W_G el tamaño de la ventana general, llevarán en el campo RI el identificador del proceso de reinicio que la fuente acaba de dar por finalizado y el flag RS a cero indicando que dicho reinicio ha terminado; este mecanismo permite a la fuente avisar al resto de sistemas de la conclusión del proceso de reinicio activo. Tras el envío de estos N paquetes, y hasta que no haya un nuevo reinicio, todos los paquetes generados llevarán un cero en el campo RI. Adicionalmente y por las razones ya vistas en el establecimiento de la conexión, también el nivel superior es avisado de la finalización del procedimiento de reinicio.

Los encaminadores y los miembros, al recibir un paquete DAT cuyo campo RI identifica el último reinicio conocido y un cero en el flag RS indicando que finalizó el proceso de reinicio, continúan con su comportamiento normal. En el caso de que se dé por finalizado el procedimiento de reinicio y un encaminador no hubiera recibido el correspondiente TS

de algún hijo, supondrá que éste ya no es su hijo y borrará en la TIB la información relativa a dicho hijo. Al dar por finalizado el procedimiento de reinicio los SRs inicializan a W_G el tamaño de las ventanas asociadas a los distintos interfaces.

Debe aclararse que los encaminadores RMNP que pertenecían al árbol de distribución y a raíz de la reconfiguración han dejado de hacerlo, no recibirán ningún paquete indicando que se ha puesto en marcha un reinicio, pero gracias al temporizador de liberación ya analizado (ver sección 5.7.4), al cabo de un cierto tiempo, liberarán los recursos asociados a la conexión.

Como otra opción de diseño se ha considerado la posibilidad de que la fuente al poner en marcha un procedimiento de reinicio, en lugar de volver a retransmitir todos los paquetes pendientes de confirmación, enviase un paquete de control que tuviera como finalidad avisar del reinicio y pedir a los miembros que volvieran a confirmar cual es el último paquete recibido en secuencia. La fuente al recibir esta información, podría empezar a retransmitir a partir del primer paquete que le faltara a algún miembro. La justificación a por qué ha sido descartada esta opción, es la misma a utilizar paquetes de datos en lugar de paquetes de control en la fase de establecimiento, y esta motivación ya fue analizada detalladamente en la sección 5.7.2.

5.9 Comunicación en grupo y control de flujo

El control de flujo es un acuerdo entre la fuente y el destino de una comunicación, para limitar el flujo de paquetes que salen de la fuente, y tiene como propósito asegurar que cada paquete que llegue al destino encontrará un buffer disponible.

Un esquema ampliamente utilizado para controlar el flujo es el de ventana deslizante, ventana que puede ser estática o dinámica.

Tradicionalmente, los esquemas de ventana utilizan una única ventana en la fuente, de modo que ésta, tiene limitado el número máximo de paquetes (o bytes) que puede tener pendientes de confirmación, estando este número máximo determinado por el tamaño de la ventana.

Los esquemas de ventana estática suponen que la capacidad para "consumir mensajes" del destino se mantiene relativamente estable mientras dura la conversación, de forma que es suficiente negociar un tamaño de ventana inicial y usarla durante todo el tiempo.

Los esquemas de ventana dinámica son más complejos, éstos permiten ir variando la ventana del transmisor como respuesta a cambios, en la capacidad del destino o en el camino que conduce al destino. Debido a su habilidad para adaptarse a variaciones en la carga, los esquemas de ventana dinámica además de desarrollar una función de control de flujo indirectamente pueden ayudar a que no se congestione la red.

El ajustar adecuadamente el tamaño de una ventana dinámica puede ser un factor decisivo en una comunicación, y sin embargo, no es una tarea trivial. Una ventana demasiado grande puede provocar que el receptor se vea saturado de paquetes, y al no disponer de buffers suficientes para almacenarlos temporalmente, tenga que optar por descartar algunos de estos paquetes; este hecho provocará posteriormente retransmisiones innecesarias. Por el contrario, una ventana demasiado pequeña tiene como consecuencia una reducción innecesaria del caudal.

Un ajuste correcto del tamaño de la ventana, además de considerar los recursos disponibles en el receptor (capacidad de proceso y buffers), debe tener en cuenta el retardo y el ancho de banda en el camino que une el emisor y el receptor. En breves líneas, a medida que aumenta el retardo conviene aumentar el tamaño de la ventana con la finalidad de no limitar innecesariamente el caudal, y a medida que aumenta el ancho de banda conviene disminuir el tamaño de la ventana con la finalidad de no saturar al receptor y provocar retransmisiones innecesarias.

Es posible generalizar los esquemas de ventana para ser usados en un entorno multipunto. Sin embargo, en dicha generalización se han identificado dos problemas:

- Dado que los *n* destinos comparten la misma ventana y, los caminos que conducen a estos *n* destinos pueden ser muy dispares en características de retardo y ancho de banda, es prácticamente imposible ajustar el tamaño de la ventana de forma que no se produzcan retransmisiones innecesarias o no haya una reducción innecesaria del caudal. Esta imposibilidad de conjugar las necesidades de los *n* destinos que comparten la misma ventana, está tanto más acentuada cuanto más dispares son, o bien los recursos disponibles en los miembros o bien las características de retardo y ancho de banda de los caminos que conducen a los dichos destinos. Obviamente a medida que aumenta el tamaño del grupo se incrementa la probabilidad de que dichas características sean dispares.

  Para minimizar la problemática planteada, en RMNP no existe una única ventana a ser compartida por todos los miembros, sino que cada sistema retransmisor asocia a cada interfaz hijo una ventana dinámica, de este modo disminuye el número de

miembros que comparten la misma ventana. El mecanismo usado para ajustar dinámicamente el tamaño de estas ventanas tiene como objetivos los ya planteados, lograr un buen caudal y al mismo tiempo no saturar al receptor con muchos paquetes, con este fin el tamaño inicial de las ventanas es alto, y posteriormente, si se producen un gran número de retransmisiones, señal implícita de que uno o más miembros no son capaces de cursar el caudal recibido, el tamaño de la ventana es reducido; este proceso de reducción continúa hasta alcanzar un punto de equilibrio.

❒ El caudal máximo está limitado por el destino más lento o por el camino a un destino con menos recursos disponibles, en los encaminadores (capacidad de proceso y memoria) o en los enlaces (ancho de banda). Esta problemática ofrece distintas soluciones:

- ♦ Agrupar los destinos en grupos disjuntos [CA94, MV95]. De forma que destinos con capacidades equivalentes estén en el mismo grupo y la fuente lleve conversaciones independientes con cada grupo. En este caso el subgrupo con mayor capacidad recibirá la información antes o recibirá la misma información pero más exhaustiva (por ejemplo vídeo de más resolución).

- ♦ Aceptar que el caudal máximo venga limitado por el miembro más lento o por el camino perteneciente al árbol de distribución con menos capacidad de transporte de paquetes, bien por disponer de menos recursos o por estar más cargado.

- ♦ Expulsar del grupo a aquellos miembros que no sean capaces de aceptar un caudal mínimo, o porque ellos mismos no disponen de los recursos suficientes o bien porque el camino en el árbol de distribución, que conduce a dicho miembro, no es capaz de cursar este caudal mínimo. La alternativa anterior es un subconjunto de ésta al solicitar un caudal mínimo de cero.

  RMNP ha adoptado la tercera solución planteada y permite que el usuario del servicio especifique un caudal mínimo a ser cursado. En el caso de que la fuente detecte que, al evolucionar el sistema para adaptarse a las necesidades de los miembros e ir decrementándose el tamaño de las ventanas, es imposible cursar el caudal mínimo, la fuente envía una indicación al resto de los sistemas retransmisores avisando que no se puede reducir más, o incluso que es necesario aumentar, el tamaño de las ventanas. Después de recibida una de las mencionadas indicaciones, si un sistema retransmisor detecta que determinados miembros no son capaces de cursar el caudal mínimo (están originando un gran número de retransmisiones), éstos son expulsados del árbol RMNP.

Hasta la actualidad han sido propuestos muchos protocolos multipunto que proporcionan distintos servicios de fiabilidad, ordenación, etc. Sin embargo un hecho a reseñar es que muy pocos de estos protocolos incorporan algún mecanismo de control de flujo. Este hecho puede deberse a que la mayoría de los protocolos que dan servicios de fiabilidad, lo hacen con aproximaciones orientadas al receptor, y al seguir este paradigma, es generalmente complejo y costoso incorporar mecanismos de control de flujo.

5.9.1 Sistema de ventanas

Los sistemas retransmisores tienen una ventana de transmisión asociada a cada interfaz hijo. Estas ventanas son dinámicas y se utilizan para controlar el flujo entre la fuente y los miembros.

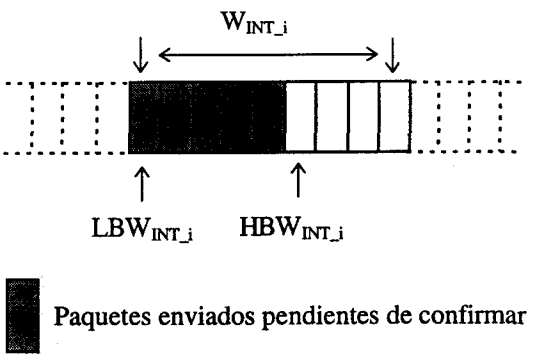


Figura 5.14: Ventana asociada al interfaz *i*

Cada ventana (ver Figura 5.14) está asociada a un determinado interfaz *i* y consta de:

- Un tamaño (W_{INT_i}).
- Un límite inferior (LBW_{INT_i}). Este límite inferior indica cual es el primer paquete (con número de secuencia más bajo) que está pendiente de ser confirmado por al menos uno de los hijos accesibles a través del interfaz *i*.
- Un límite superior (HBW_{INT_i}). Este límite superior indica cual es el siguiente nuevo paquete a distribuir por el interfaz *i*.

La fuente además de mantener una ventana por cada interfaz hijo, mantiene una *ventana general*. Esta ventana es estática estando su tamaño definido por el parámetro de configuración denominado *Tamaño de la ventana general* (W_G). Con el fin de evitar la duplicación de paquetes (que una entidad RMNP en un miembro, entregue el mismo paquete más de una vez al usuario del servicio), el tamaño de la ventana general debe ser como máximo la mitad del espacio de números de secuencia. De aquí que W_G deba tomar un valor en el rango $[1, 2^{23}]$[18].

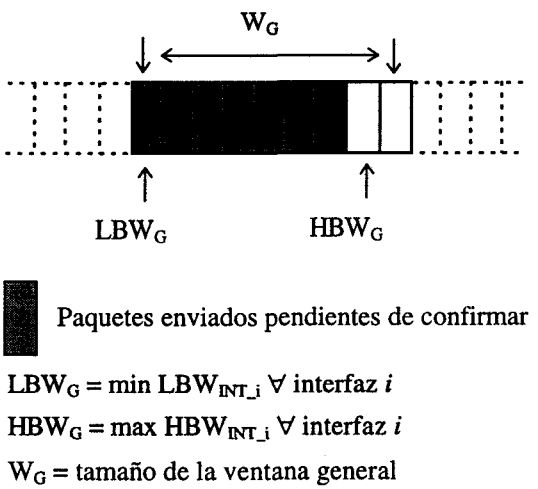


Figura 5.15: Ventana general

La *ventana general* (ver Figura 5.15) también tiene asociados un limite inferior (LBW_G), que será el mínimo de los limites inferiores de las ventanas asociadas a los interfaces hijo en la fuente, y un límite superior (HBW_G) que será el máximo de los límites superiores de las ventanas asociadas a los interfaces hijo en la fuente.

A continuación se verá cómo se actualizan los límites superior e inferior de las ventanas asociadas a los interfaces, para posteriormente exponer cómo se actualizan los límites superior e inferior de la ventana general.

Cada vez que un sistema retransmisor recibe un ACK_*n*, desde un hijo accesible a través del interfaz *i*, comprueba si es posible desplazar el límite inferior de la ventana. Esto será posible si con esta confirmación recibida se cumple que, todos los hijos accesibles a

[18] El espacio de números de secuencia de RMNP es 2^{24}

través de *i* ya han confirmado hasta el paquete *m* (siendo m ≤ n), en cuyo caso, el sistema retransmisor desplaza el límite inferior de la ventana actualizando LBW_{INT_i} a *m+1*.

Cada vez que un sistema retransmisor recibe un nuevo paquete de datos *n* a ser distribuido (en la fuente por habérselo entregado el usuario del servicio o en un SR por haberlo recibido desde su padre en secuencia), para cada interfaz hijo *i* hace lo siguiente:

- Comprueba si $HBW_{INT\_i} - LBW_{INT\_i} = W_{INT\_i}$, en cuyo caso, retarda el envío del paquete por el interfaz *i* hasta que desaparezca esta condición.

- En el caso de que $HBW_{INT\_i} - LBW_{INT\_i} < W_{INT\_i}$, el SR distribuye el paquete por el interfaz y, desplaza el límite superior de la ventana, incrementando en uno HBW_{INT_i}.

Como ya se ha dicho, la fuente además de disponer de una ventana asociada a cada interfaz hijo, que será gestionada según se acaba de ver, también usa y mantiene una ventana general. La fuente usa esta ventana general cada vez que quiere distribuir un nuevo paquete, de modo que lo primero que comprueba es si $HBW_G - LBW_G = W_G$, en cuyo caso retarda la distribución de este paquete hasta que desaparezca esta condición. Por otro lado, cada vez que la fuente actualiza uno de los límites de alguna ventana asociada a un interfaz, comprueba si como consecuencia también debe actualizar los límites de la ventana general, ya que el límite inferior y superior de la ventana general vienen marcados por el mínimo y el máximo respectivamente de los límites inferiores y superiores de las ventanas asociadas a los interfaces hijo en la fuente.

5.9.2 Mecanismo de control de flujo en RMNP

En la fase de establecimiento (ver sección 5.7.2) únicamente la fuente tiene ventanas asociadas a los distintos interfaces y su tamaño viene definido por el parámetro W_G.

Posteriormente, al comenzar la fase de transferencia los SRs también inicializan el tamaño de las ventanas asociadas a los interfaces a W_G. Dichas ventanas tienen un tamaño máximo de W_G y un tamaño mínimo de uno; los sistemas retransmisores varían el tamaño de éstas de acuerdo con el número de retransmisiones que se están produciendo por el correspondiente interfaz; con este fin, llevan cuenta del número de veces que han retransmitido cada paquete pendiente de confirmación (siendo la retransmisión, por tempori-

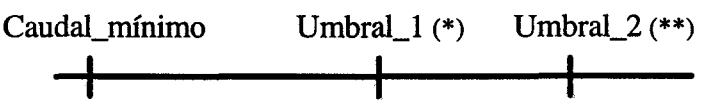
zación o por solicitud explícita). Si se detecta que por un interfaz se están realizando muchas retransmisiones el sistema hace lo siguiente:

1. Si la fuente tiene permitido reducir el tamaño de las ventanas, el sistema reduce la ventana del correspondiente interfaz.

2. Si la fuente no tiene permitido reducir el tamaño de las ventanas porque el caudal es un factor crítico, el sistema pone en marcha una liberación parcial, expulsando del árbol RMNP al hijo o hijos que han causado ese número excesivo de retransmisiones.

El usuario del servicio, al establecer la conexión, indica en Kbytes/sg cual es el caudal mínimo que precisa cursar. Este caudal mínimo puede ser cero si la aplicación no tiene ningún inconveniente en adaptarse a la velocidad del miembro más lento.

Durante toda la fase de transferencia la fuente mantiene una medida del caudal medio cursado (denominado *caudal_cursado*). La fuente obtiene una nueva medida del caudal_cursado cada cierto tiempo. Este periodo de tiempo no es fijo, hecho que podría dar lugar a inconsistencias debido a que el flujo de entrega de datos del usuario del servicio puede no ser uniforme, sino que la fuente calcula una nueva medida del caudal_cursado cada vez que ha mandado y recibido confirmación de $2W_G$ nuevos paquetes, o lo que es lo mismo, la ventana general se ha desplazado en $2W_G$ posiciones. Por ejemplo, si W_G=8 y la fase de transferencia se ha iniciado con el paquete con número de secuencia 16, la fuente realiza la primera medida del caudal cursado al recibir confirmación del paquete con número de secuencia 32, la siguiente al recibir confirmación del paquete con número de secuencia 48 y así sucesivamente. Para calcular en estos puntos el caudal_cursado, la fuente necesita medir el tiempo que se ha tardado en mandar y recibir confirmación de los últimos $2W_G$ paquetes, así como los bytes enviados en este tiempo.



(*) Umbral_1= Caudal_mínimo * φ
(**) Umbral_2= Caudal_mínimo * ω

Figura 5.16: Umbrales de caudal

La fuente además de conocer el caudal mínimo mantiene ciertos umbrales de caudal (ver Figura 5.16) que le permiten detectar si el caudal_cursado está acercándose demasiado al caudal mínimo, y antes de alcanzar este límite tomar las acciones correctivas oportunas. En grandes líneas, estas acciones correctivas consisten en que la fuente impedirá en mayor o menor grado la reducción del tamaño de las ventanas, hecho que produciría una disminución del caudal_cursado. La fuente utiliza unos flags existentes en la cabecera de los paquetes de datos (denominados en conjunto flags de *gestión de ventanas*), para indicar a los SRs si está permitido, y en que grado, reducir el tamaño de las ventanas. Más en concreto, según en qué rango esté la última medida del caudal_cursado la fuente hace lo siguiente:

- Si el caudal_cursado está por encima del Umbral_2, la fuente permite que las ventanas sean reducidas. Esto es indicado de forma implícita a los SRs no activando ningún flag de gestión de ventanas.

- Si está entre el Umbral_1 y el Umbral_2, la fuente sólo permite que las ventanas sean reducidas de forma provisional. Esto es indicado a los SRs activando el flag *Reducciones provisionales* de todos los paquetes de datos enviados, hasta que se vuelva a calcular una nueva medida del caudal cursado.

- Si está entre el caudal mínimo y el Umbral_1, la fuente no permite que se realice ninguna reducción de ventana, y al mismo tiempo solicita que se incremente el tamaño de las ventanas cuyo tamaño está reducido provisionalmente. Esto es indicado a los SRs activando el flag *Prohibido reducir* de todos los paquetes de datos enviados, hasta que se vuelva a calcular una nueva medida del caudal cursado.

- Si dos medidas consecutivas del caudal_cursado están por debajo del caudal mínimo, la fuente solicita que el tamaño de todas las ventanas sea incrementado y ella misma lo hace en sus interfaces. Esto es indicado a los SRs activando el flag *Incrementar ventana* del primer paquete de datos enviado, tras obtener por segunda vez una medida del caudal_cursado que está por debajo del caudal mínimo; el resto de los paquetes, hasta obtener dos nuevas medidas del caudal_cursado, llevarán activo el flag *Prohibido reducir*. El hecho de solicitar el incremento de las ventanas cuando hay dos medidas sucesivas del caudal_cursado por debajo del caudal mínimo en lugar de sólo una, se debe a lo siguiente: (1) una caída brusca del caudal puede ser originada por una situación transitoria, y no es deseable que debido a una situación transitoria sean expulsados del árbol RMNP los miembros que no sean capaces de soportar el incremento de la ventana; (2) una vez solicitado un aumento de la ventana es necesario esperar a que se tomen al menos dos nuevas medidas del cau-

dal_cursado, para poder analizar los efectos producidos por esta acción correctiva, antes de que la fuente decida de nuevo si es necesario aumentar aún más el tamaño de las ventanas.

A continuación se expone en más detalle como los sistemas retransmisores van variando el tamaño de la ventana asociada a cada interfaz. En todo momento la fuente conoce si es posible o no reducir el tamaño de las ventanas, pues ella misma lo decide en función del caudal_cursado, en el caso de los SRs, éstos recuerdan cual ha sido la última indicación de la fuente, contenida en el último paquete de datos recibido en secuencia, de modo que si este último paquete: (1) no llevaba ningún flag de gestión de ventanas activo, sabe que está permitida la reducción del tamaño de las ventanas; (2) llevaba activo el flag *Reducciones provisionales,* sabe que puede reducir el tamaño de la ventana pero de modo provisional; (3) llevaba activo el flag *Prohibido reducir* o *Incrementar ventana,* sabe que no puede reducir el tamaño de ninguna ventana.

Un SR al recibir un paquete de datos con el flag *Incrementar ventana* activado, responde ante esta indicación de la fuente, incrementando la ventana de todos los interfaces, de modo que el nuevo tamaño de la ventana será: max $[1+W_{INT\_i}, \gamma^{(19)} * W_{INT\_i}]$ siendo $\gamma > 1$, obviamente si W_{INT_i} tenía el valor máximo, se mantiene inalterable. La fuente al detectar que deben incrementarse las ventanas tendrá este mismo comportamiento con sus propios interfaces hijo.

Los sistemas retransmisores, además de conocer si está permitido o no reducir el tamaño de las ventanas, mantienen asociado a cada interfaz hijo el número de veces que ha sido retransmitido cada paquete pendiente de confirmación, bien haya sido retransmitido por solicitud explícita o por el vencimiento de un temporizador. Si el número de retransmisiones de un paquete *n* en el interfaz *i* alcanza el valor del parámetro *Número máximo de veces que puede retransmitirse un paquete antes de reducir la ventana* (MAX_RET_BWD), el comportamiento del sistema retransmisor dependerá de si está permitido o no reducir el tamaño de las ventanas:

- Si está permitido reducir el tamaño de las ventanas, el sistema reduce el tamaño de la ventana asociada al interfaz *i* (W_{INT_i}), de modo que el nuevo tamaño de la ventana será min $[W_{INT\_i}-1, \beta^{(20)} * W_{INT\_i}]$ siendo $\beta < 1$, obviamente si W_{INT_i} tenía el valor mínimo, se mantiene inalterable.

[19] Parámetro de configuración.

[20] Parámetro de configuración.

❒ Si está prohibido reducir el tamaño de las ventanas, el sistema comprueba que hijo o hijos de los accesibles por el interfaz *i*, no han confirmado todavía el paquete *n* y para cada uno de éstos hace lo siguiente:

- ♦ Borra la información de estado asociada al hijo y le elimina de la TIB.

- ♦ Envía al hijo un paquete de liberación con el flag AB activado, especificando la causa de la liberación (imposibilidad de cursar el caudal mínimo). Este LIB se envía como un paquete punto a punto del padre al hijo. Dicho hijo se encargará de distribuirlo a sus correspondientes hijos hasta que llegue a los miembros.

  No es preciso asociar un temporizador al paquete LIB pues en el caso de que se perdiera, en los sistemas afectados se vencería el temporizador T_{LIB} (ver sección 5.7.4) y éstos detectarían el aborto de la conexión.

- ♦ Envía a la fuente un paquete LIB, con el flag INF activado, especificando la causa de la liberación. Seguidamente, activa el temporizador T_{WS} mientras espera que la fuente le confirme la recepción de este paquete, mediante un LIB_ACK con el correspondiente flag INF activado. La fuente al recibir este paquete LIB informa de lo acontecido al nivel superior.

- ♦ Comprueba si tras la expulsión del hijo continúa teniendo otros hijos y en caso negativo solicita a su padre salir del árbol.

- ♦ Comprueba si el resto de los hijos ya habían confirmado el paquete *n* en cuyo caso envía un ACK a su padre.

❒ Si está permitido reducir el tamaño de las ventanas de forma provisional, el sistema reduce el tamaño de la ventana asociada al interfaz *i* (W_{INT_i}), de modo que el nuevo tamaño de la ventana será min [W_{INT_i}-1, α*W_{INT_i}] siendo α[21] < 1, obviamente si W_{INT_i} tenía el valor mínimo, se mantiene inalterable.

El permitir reducciones provisionales se debe al hecho de que aunque el caudal cursado esté relativamente cerca del caudal mínimo, existe la posibilidad de que el miembro que realmente está limitando el caudal esté accesible por otro camino distinto y el disminuir esta ventana no reducirá el caudal cursado. Un sistema re-

[21] Parámetro de configuración.

transmisor después de reducir de manera provisional el tamaño de una ventana se mantiene a la espera de recibir una indicación:

- Si de nuevo las retransmisiones llegan a MAX_RET_BWD, el sistema realiza en el interfaz *i* una liberación parcial.

- Si desde que se realizó la reducción provisional la ventana asociada al interfaz *i* se ha desplazado en $3W_G$ posiciones y no se ha recibido ningún paquete desde la fuente, indicando que está prohibido reducir el tamaño de las ventanas, el sistema da la reducción por definitiva.

- Si recibe un paquete de datos indicando que está prohibido reducir el tamaño de las ventanas, hecho que significará que ésta u otra reducción provisional ha reducido aún más el caudal_cursado, devuelve a la ventana su tamaño previo. Si seguidamente se vuelve alcanzar de nuevo MAX_RET_BWD, estando prohibido reducir las ventanas, el sistema realiza una liberación parcial en el interfaz *i*.

Como puede observarse, los sistemas retransmisores empiezan dando a las ventanas un tamaño máximo que va reduciéndose al detectarse muchas retransmisiones y sin embargo aparte del aviso de la fuente de que es necesario aumentar el caudal para satisfacer las necesidades del usuario del servicio, no se propone ningún otro mecanismo para aumentar el tamaño de las ventanas, esto tiene como justificación que si el caudal cursado es aceptable para el usuario del servicio, no es deseable cargar más la red aumentando del tamaño de las ventanas.

RMNP no propone ningún mecanismo específico para el control de la congestión, que permita controlar la carga de la red antes de que ésta empiece a perder paquetes, dado que presupone que esta función es realizada por el nivel inferior (servicio de distribución de datagramas multipunto). Nótese que es más adecuado realizar esta función en el nivel inferior, dado que el número de encaminadores RMNP en la interred puede ser muy pequeño y esta función de control de congestión es una función global de la interred que debe implicar al mayor número posible de nodos.

Aunque RMNP no propone ningún mecanismo específico para controlar la congestión, debe observarse que el mecanismo de control de flujo propuesto indirectamente ayuda a que no se congestione la red, esto es así dado que el aumento en el número de retransmisiones puede deberse a que un miembro está siendo saturado con paquetes y no dispone de buffers suficientes para ir almacenando los paquetes recibidos, pero también podría estar causado por una congestión más o menos grave en la red. La respuesta de RMNP al au-

mentar el número de retransmisiones es reducir la ventana asociada, hecho que ayuda a controlar la congestión si ésta era la causa del excesivo número de retransmisiones, o al menos impide que la red se congestione aún más.

5.10 Ajuste de temporizadores

En esta sección se analiza el problema de a qué valores y cómo, de forma estática o dinámica, ajustar los distintos temporizadores usados en los sistemas RMNP, con este fin se recuerda la motivación de cada uno de los temporizadores y se plantean las problemáticas derivadas de hacer un incorrecto ajuste de éstos.

5.10.1 Temporizador de retransmisión.

Este temporizador es utilizado en la fuente y en los SRs. Cada vez que uno de estos sistemas envía por un interfaz un paquete de datos, le asocia un temporizador de retransmisión (de aquí en adelante denominado T_{RET}). Si un T_{RET} expira, sin haberse recibido el asentimiento correspondiente, se asume que el paquete asociado se ha perdido, volviéndose a transmitir por el correspondiente interfaz.

Los temporizadores de retransmisión tienen asociados un mecanismo de ajuste adaptativo. Este mecanismo permite calcular de forma dinámica el valor a asignar a cada temporizador de retransmisión en el momento de ser activado.

Primeramente se analiza por qué es adecuado utilizar un mecanismo de ajuste dinámico y los distintos factores que condicionan su diseño, para posteriormente detallar el mecanismo de ajuste usado por RMNP.

El RMNP ha sido diseñado para trabajar en una interred que puede ser muy extensa, y donde los miembros pueden estar conectados en cualquier punto de ésta; dado esto, es más adecuado utilizar un mecanismo de ajuste adaptativo, que asignar a todos los temporizadores de retransmisión activados un mismo valor. Esto se debe a los siguientes motivos:

- La posibilidad de que unos miembros estén muy próximos al punto de retransmisión (fuente o SR) y otros muy alejados, hace que sea imposible conocer a priori en cuánto tiempo, desde que fue enviado el paquete de datos, debería llegar el asentimiento correspondiente.

- El retardo que sufre cada paquete en la interred depende del tráfico existente, como consecuencia, dependiendo del momento, puede variar dramáticamente el tiempo total necesario para que un paquete se distribuya a los destinos y vuelva el asentimiento correspondiente.

Al diseñar un mecanismo de ajuste deben considerarse los siguientes factores:

- Disponer de unos temporizadores de retransmisión muy ajustados puede originar retransmisiones innecesarias. Por ejemplo, podría retransmitirse un paquete cuando lo único que ha ocurrido es que el asentimiento no ha llegado debido a un pequeño aumento en el retardo de la red. Además debe observarse que en el caso de que dicho aumento en el retardo de tránsito en la red, sea debido a una situación de congestión incipiente, ésta puede agravarse debido a las retransmisiones innecesarias.

- Disponer de unos temporizadores de retransmisión con valores muy altos puede originar una pérdida en la eficiencia del protocolo. Debe observarse que en el caso de que haya una pérdida de paquetes, el temporizador tardará mucho tiempo en expirar, con la consiguiente disminución en el caudal del protocolo. Aunque esta afirmación es, en general, cierta para todos los protocolos que utilizan temporizadores de retransmisión para detectar la pérdida de paquetes, RMNP tiene la peculiaridad de disponer de mecanismos explícitos para solicitar la retransmisión de ciertos paquetes. De este modo, lo normal será que la retransmisión de paquetes perdidos se origine por una petición explícita por parte del miembro o miembros afectados. Mientras que únicamente cuando se pierda un paquete que solicita una retransmisión explícitamente o se pierdan los n últimos paquetes de datos de una ráfaga, la retransmisión será originada por el vencimiento de un temporizador.

- En RMNP se guarda una copia de cada paquete pendiente de confirmación en la fuente y en los SRs.

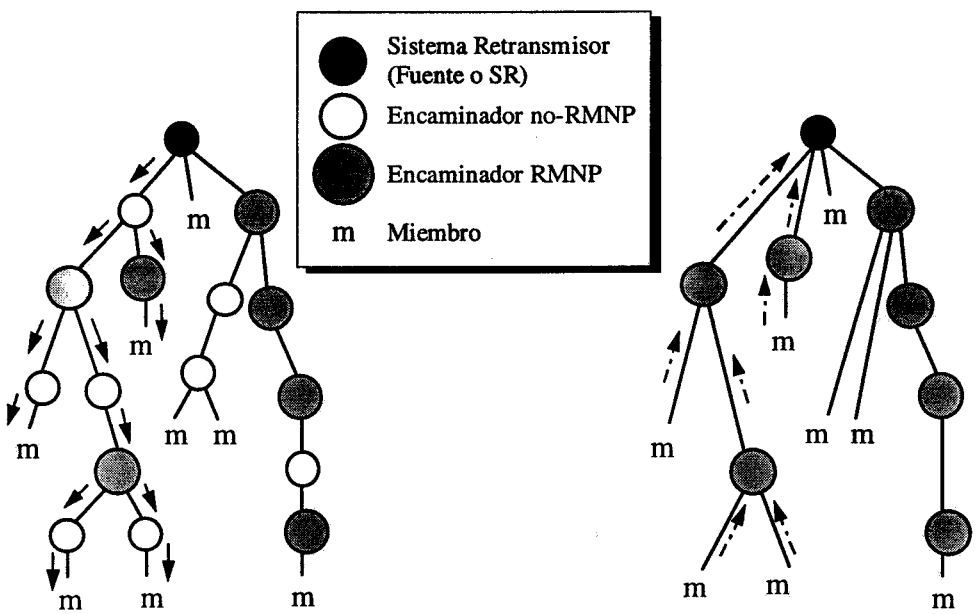
De lo anterior se deduce que:

- Es necesario mantener un cierto equilibrio entre los dos extremos, temporizadores demasiado ajustados y demasiado bondadosos.

- En la mayoría de los casos las retransmisiones vendrán originadas, por una solicitud explícita del miembro o miembros afectados. De aquí que en RMNP será más conveniente disponer de temporizadores de retransmisión con valores relativamente altos.

- Debe favorecerse el hecho de que la retransmisión de los paquetes perdidos se haga desde el sistema retransmisor más cercano al punto donde se perdió el paquete. Este comportamiento tiene dos ventajas: (1) los paquetes retransmitidos llegarán antes a los destinos, disminuyéndose el retardo medio de distribución y, (2) el paquete sólo se volverá a distribuir por una rama del subárbol con raíz en dicho sistema retransmisor más cercano, con el consiguiente ahorro en proceso y ancho de banda.

El mecanismo de ajuste de T_{RET} usado en RMNP, es una generalización del utilizado en la mayoría de las implementaciones de TCP [Jac88]. Esto se debe principalmente a dos razones: (1) dicho mecanismo está ampliamente probado y testeado, habiéndose obtenido unos resultados satisfactorios con su uso, y (2) es posible hacer de forma sencilla, una generalización de este algoritmo para adaptarlo al entorno de comunicaciones multipunto. En otros protocolos ya se han hecho diversas modificaciones de este mecanismo para adaptarlo al entorno multipunto [LS96, WMK94].



a) Tiempo de distribución

b) Tiempo de agregación de los asentimientos

Nota: Los paquetes de datos se propagan siguiendo el árbol de distribución mientras que los asentimientos se mandan como mensajes unicast desde cada hijo RMNP a su correspondiente padre.

Figura 5.17: Cálculo del IRTT

El mecanismo de ajuste de los temporizadores en TCP está basado en el RTT[22], siguiendo la misma filosofía y teniendo en consideración que en RMNP los temporizadores de retransmisión están asociados a interfaces, RMNP estima **el tiempo de ida y vuelta por una rama de un subárbol**, o lo que es lo mismo, **el tiempo de ida y vuelta por un interfaz** (IRTT). El IRTT de un paquete en un interfaz de un sistema retransmisor (SR o fuente) se define como, la suma del *tiempo de distribución del paquete* y del *tiempo de agregación de los asentimientos asociados a dicho paquete* (ver Figura 5.17). El tiempo de distribución del paquete es, el tiempo que se necesita para distribuir dicho paquete a todos los descendientes RMNP accesibles por el mencionado interfaz. Mientras que el tiempo de agregación de los asentimientos es, el tiempo que los asentimientos correspondientes tardan en ser agregados hasta llegar al sistema retransmisor. Debe observarse que el IRTT vendrá determinado por el miembro, de entre los accesibles por el interfaz, más alejado en tiempo del sistema retransmisor.

Al igual que TCP, RMNP usa los siguientes algoritmos para ajustar el temporizador de retransmisión (basándolos en el IRTT):

- El *algoritmo de Jacobson* [Jac88], permite ajustar T_{RET} a partir de una media y una varianza estimadas de IRTT.

- El *algoritmo de Karn* [KP87], selecciona las medidas del IRTT que no son ambiguas, y consiguientemente son adecuadas para ser utilizadas en el cálculo de una estimación de la media y la varianza de IRTT.

- La estrategia del *incremento exponencial binario*, establece qué valores de temporizador deben ser asociados a retransmisiones sucesivas del mismo paquete.

A continuación, cada uno de los anteriores algoritmos se expone con más detalle.

5.10.1.1 El algoritmo de Jacobson.

Cada sistema retransmisor va tomando continuamente muestras de IRTT y usa éstas para calcular un valor estimado a la media y la varianza de IRTT. El algoritmo de Jacobson calcula el valor a asignar a T_{RET} en base a estos valores estimados para la media y la varianza. El calcular un valor para T_{RET} no utilizando únicamente la media estimada del IRTT sino su variación en el tiempo, se debe al hecho de que se ha observado que cuando

[22] *Round Trip Time.*

determinados caminos se congestionan, la variación en el retardo que sufre un paquete es muy grande [Jac88].

De acuerdo con lo comentado, el algoritmo de Jacobson propone que el cálculo del valor del temporizador de retransmisión puede hacerse de forma efectiva usando las siguientes fórmulas:

$$\begin{aligned} Err &= M_{IRTT} - A_{IRTT} \\ A_{IRTT} &= A_{IRTT} + g_A(Err) \\ D_{IRTT} &= D_{IRTT} + g_D(|Err| - D_{IRTT}) \\ T_{RET} &= A_{IRTT} + 4D_{IRTT} \end{aligned} \qquad \textbf{(5.1)}$$

Siendo,

g_A y g_D: términos de ganancia
M_{IRTT}: una nueva muestra del IRTT
A_{IRTT}: la media estimada del IRTT
D_{IRTT}: la desviación estimada del IRTT

La experimentación ha mostrado que 0,0625 y 0,125 son buenos valores para g_A y g_D respectivamente [WMK94].

Al comentar los objetivos de diseño del mecanismo de ajuste de temporizadores, ya se expusieron las razones por las que en RMNP es más conveniente disponer de temporizadores de retransmisión con valores relativamente altos. Para conseguir este objetivo de diseño, la expresión (1) del algoritmo de Jacobson ha sido sustituida en RMNP por:

$$T_{RET} = 2(A_{IRTT} + 4D_{IRTT}) \qquad \textbf{(5.2)}$$

Es importante señalar que el hecho de basar el cálculo de los valores para T_{RET} en IRTT tiene dos ventajas: (1) permite utilizar la misma filosofía y los mismos algoritmos tanto en la fuente como en los SRs; (2) hace que los valores asignados a T_{RET} en cada uno de los sistemas retransmisores sea mayor a medida que están más lejanos en tiempo de los miembros, y con esto, se consigue el objetivo de primar las retransmisiones, desde el sistema retransmisor más cercano al punto donde se originó el problema.

5.10.1.2 El algoritmo de Karn.

En teoría medir una muestra del IRTT es trivial, consiste en restar del instante en el que llegó el último asentimiento a un paquete de datos, el instante en el que fue enviado dicho paquete. Sin embargo, en el caso de que un paquete sea retransmitido por haberse vencido el temporizador, cuando posteriormente llegue el asentimiento a este paquete, no habrá ninguna forma de saber si dicho asentimiento corresponde al paquete original o al retransmitido. Este fenómeno se denomina el de la ambigüedad de los asentimientos.

En [Com91] se analiza cómo ninguna de las dos aproximaciones posibles, asociar el asentimiento con la transmisión original o asociarlo con la retransmisión más reciente, funciona correctamente.

Para resolver esta problemática Karn [KP87] propone un algoritmo muy simple: no se debe actualizar el IRTT estimado con muestras obtenidas a partir de paquetes que han sido retransmitidos, o lo que es lo mismo, sólo se actualizará el IRTT estimado con muestras del IRTT obtenidas a partir de asentimientos no ambiguos, asentimientos asociados a paquetes que han sido transmitidos una única vez.

5.10.1.3 Estrategia de incremento exponencial binario.

Una implementación simplista del algoritmo de Karn que ignore los asentimientos ambiguos para recalcular el IRTT estimado puede conducir a fallos. Por ejemplo, supóngase que se envía un paquete después de haberse producido un incremento brusco en el retardo, según lo visto, el valor para T_{RET} se calculará usando las medidas estimadas de IRTT existentes. Debido al incremento brusco en el retardo, el valor calculado y asignado a T_{RET} será demasiado pequeño y forzará una retransmisión. Cuando llegue el asentimiento correspondiente será ignorado al ser considerado ambiguo, como resultado de este comportamiento, nunca se actualizará el IRTT estimado y cada nuevo valor calculado para T_{RET} será demasiado pequeño, forzando una o varias retransmisiones.

Para evitar este comportamiento erróneo es preciso combinar los algoritmos ya vistos con una estrategia de incremento exponencial. Una estrategia de este tipo hace que el valor asignado a T_{RET}, la primera vez que se activa para cada paquete, se calcule por la expresión (2), mostrada antes; pero sin embargo, cada vez que el temporizador expira causando una retransmisión, se dobla el valor de T_{RET}:

$$T_{RET} = 2^i T_{RET} \qquad \textbf{(5.3)}$$

Siendo,

i: número de retransmisiones del mismo paquete.

Al combinar el algoritmo de Karn con una estrategia de incremento exponencial binario, el comportamiento resultante es el siguiente: cuando la interred tiene un comportamiento extraño (un aumento brusco en el retardo), el algoritmo de Karn calcula los valores para T_{RET} sin utilizar los valores estimados para IRTT. Esto significa que se usa el IRTT estimado para calcular un valor inicial a T_{RET}, pero a partir de la primera retransmisión y hasta que se logra transferir con éxito un paquete (sin ser precisa una retransmisión), los distintos valores asignados a T_{RET} van incrementándose exponencialmente; además para cada nuevo paquete se conserva el valor de temporizador que resultó del incremento exponencial. Finalmente, cuando llega un asentimiento asociado a un paquete que no precisó retransmisión (asentimiento no ambiguo), se recalcula el IRTT y utilizando éste se calcula el siguiente valor para T_{RET}.

Debe notarse que en un principio, por ejemplo en el caso de la fuente en la fase de establecimiento, los sistemas retransmisores no habrán obtenido muestras válidas de IRTT, y por lo tanto, no dispondrán de datos para calcular un valor inicial a T_{RET} según (2). En estas circunstancias el valor asignado a T_{RET} será el indicado por el parámetro de configuración T_{RET0}.

Como una alternativa de diseño se ha estudiado la posibilidad de disponer de temporizadores de retransmisión, así como sus correspondientes mecanismos de ajuste, asociados a cada hijo, en lugar de tener uno genérico para todos los hijos. El elegir esta opción supondría aumentar la complejidad de los procedimientos utilizados, e incrementar los recursos necesarios en los sistemas retransmisores; esto podría ser justificable, si todas las recuperaciones de los paquetes perdidos se hicieran por el vencimiento de temporizadores, pero como ya se ha visto, éste no es el caso del RMNP.

5.10.1.4 T_{RET} y detección de determinadas situaciones anómalas

En el transcurso de una comunicación multipunto son posibles ciertas situaciones anómalas, tales como que:

- ❒ Se produce una partición en la interred.
- ❒ Un miembro deja el grupo, sin notificar el hecho.
- ❒ Un miembro falla.

Planteada una de estas situaciones anómalas, es deseable que no afecte a todo el grupo sino únicamente al miembro causante de (por ejemplo, un miembro que salió del grupo sin notificarlo) o, al miembro o miembros directamente afectado(s) por (una partición en la interred por ejemplo).

RMNP incorpora una serie de mecanismos que tienen como finalidad detectar tales situaciones anómalas, y evitar que afecten a miembros no implicados directamente. Estos mecanismos están basados en la detección de un número excesivo de, retransmisiones por temporizador (en los BRs) o vencimientos del temporizador de retransmisión (en los SRs). Debe observarse que el efecto indirecto provocado por las situaciones anómalas planteadas es que el sistema o sistemas afectados dejarán de confirmar paquetes, hecho que provocará que se dispare el número de vencimientos de temporizador y consiguientemente de retransmisiones por temporizador.

A continuación se analiza como los distintos sistemas detectan tales situaciones y las medidas adoptadas en su caso. En primer lugar se verá el comportamiento de los sistemas retransmisores (fuente y SRs), para posteriormente pasar a ver el de los BRs.

Cada **sistema retransmisor,** para cada uno de los interfaces hijo, lleva cuenta del número de veces que ha expirado el T_{RET} asociado a cada paquete pendiente de confirmación. Así mismo, para cada hijo h_i mantiene el número máximo de veces que está permitido que expire el T_{RET} asociado a un paquete, antes de expulsar del árbol a dicho hijo ($MAX\_RET\_BCR\_h_i$).

Dado un sistema retransmisor, en el caso de que en un determinado interfaz i se cumplan las dos siguientes condiciones: (1) h_i es un hijo asociado al mencionado interfaz que

aún no ha confirmado el paquete *n* y, (2) el número de veces que ha expirado el T_{RET} asociado a *n* alcanza MAX_RET_BCR_h_i, dicho sistema retransmisor hace lo siguiente:

- Borra la información de estado asociada a h_i y le elimina de la TIB.
- Envía a h_i un paquete de liberación con el flag AB activado, especificando la causa de la liberación (alcanzado el número máximo de retransmisiones permitidas). Este LIB se envía como un paquete punto a punto del padre al hijo. Dicho hijo se encargará de distribuirlo a sus correspondientes hijos hasta que llegue a los miembros.

  No es preciso asociar un temporizador a este paquete LIB, pues en el caso de que éste se perdiera, en los sistemas afectados se vencería el temporizador T_{LIB} (ver sección 5.7.4) y éstos detectarían el aborto de la conexión.
- Envía a la fuente un paquete LIB, con el flag INF activado, especificando la causa de la liberación. Seguidamente, activa el temporizador T_{WS} mientras espera que la fuente le confirme la recepción de este paquete, mediante un LIB_ACK con el correspondiente flag INF activado. La fuente al recibir este paquete LIB informa de lo acontecido al nivel superior.
- Comprueba si el resto de los hijos ya habían confirmado el paquete *n*, en cuyo caso, envía un ACK a su padre.
- Comprueba si tras la expulsión del hijo continúa teniendo otros hijos, y en caso negativo, solicita a su padre salir del árbol.

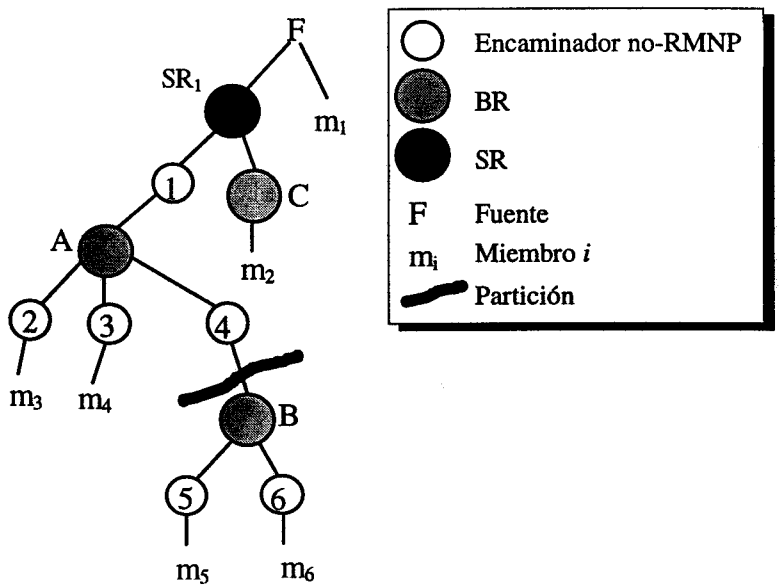


Figura 5.18: Partición en la interred

Los **BRs** no disponen de temporizadores de retransmisión por lo cual no pueden llevar cuenta del número de veces que éstos han vencido. Sin embargo, el hecho de detectar las anomalías planteadas únicamente en los sistemas retransmisores, puede provocar que en el caso de que el sistema más cercano al punto donde se originó el problema sea un BR, uno o más miembros sean expulsados del grupo de forma injusta (en el ejemplo de la Figura 5.18, la partición se detectaría en SR_1, sistema retransmisor más cercano, y éste expulsaría del grupo a m_2,m_3,m_4,m_5 y m_6). Con el fin de minimizar estas injusticias, es preciso incorporar también en los BRs un mecanismo de detección de este tipo de situaciones, y hacer que el primer sistema que detecte la anomalía, sea el más cercano al punto donde se originó el problema (en el ejemplo de la Figura 5.18, debería detectarse en *A*).

El mecanismo utilizado en los BRs es análogo al visto para los sistemas retransmisores, pero en este caso, cada BR mantiene, para cada uno de los interfaces hijo y por cada paquete pendiente de retransmisión, el número de veces que éste ha sido retransmitido sin que existiera ninguna solicitud de retransmisión pendiente, lo cual significa que han sido retransmisiones originadas por temporizador. Así mismo y de forma análoga a los sistemas retransmisores, cada hijo h_i tiene asociado un número máximo de retransmisiones por temporizador del mismo paquete ($MAX\_RET\_BCR\_h_i$), antes de sospechar que se ha producido una situación anómala y expulsarlo del árbol RMNP.

Dado un BR, en el caso de que en un determinado interfaz *i* se cumplan las dos siguientes condiciones: (1) *hi* es un hijo asociado al mencionado interfaz que aún no ha confirmado el paquete *n* y, (2) el número de retransmisiones por temporizador de un determinado paquete *n* alcanza $MAX\_RET\_BCR\_h_i$, el comportamiento del BR será idéntico al visto para los sistemas retransmisores.

5.10.1.4.1 Ajuste de $MAX\_RET\_BCR\_h_i$

Debe observarse, que de acuerdo con el comportamiento expuesto para los BRs y los SRs, en el caso de que $MAX\_RET\_BCR\_h_i$ tuviera el mismo valor en todos los sistemas padre, lo normal sería que el primer sistema en detectar la situación anómala fuera un sistema alejado del punto donde se originó el problema (partición, caída del miembro), y esto tendría como consecuencia, que uno o más miembros serían injustamente expulsados del árbol RMNP. Por ejemplo en la Figura 5.19, si la situación anómala fuera detectada en SR_2 serían expulsados del grupo m_9, m_{10} y m_{11}, si fuera detectada en D serían expulsados m_{10} y m_{11}, mientras que si fuera detectada en E únicamente sería expulsado m_{11}.

Como conclusión, para evitar en lo posible la expulsión injusta de miembros, es preciso que el primer sistema que detecte una anomalía sea el más cercano al punto donde ésta se originó. Para lograr este objetivo RMNP basa el ajuste de MAX_RET_BCR_h_i (para el hijo h_i), en el número de saltos RMNP al miembro más alejado de entre los que están accesibles a través de h_i. En el ejemplo de la Figura 5.19, para el encaminador SR_1 se tiene que el miembro más alejado de los que están accesibles a través del hijo C, es m_2 y está a 2 saltos RMNP, y los miembros más alejados de los que están accesibles a través del hijo A son m_5 y m_6, y están a 3 saltos RMNP.

Los ACKs y los NAKs incluyen un campo (denominado HFM), que sirve para que los sistemas aprendan a cuántos saltos RMNP está el miembro más alejado de los accesibles a través de cada hijo.

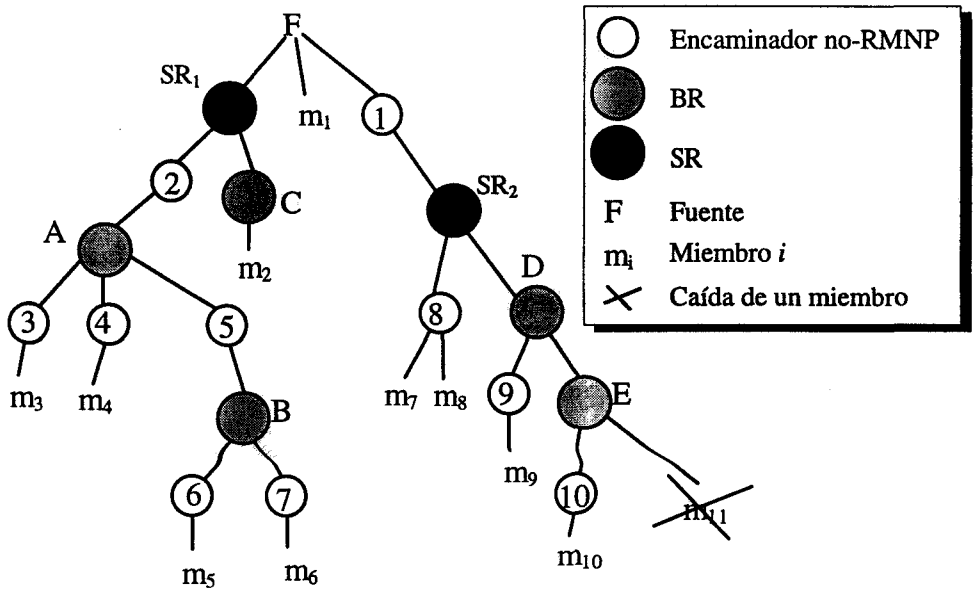


Figura 5.19: Caída de un miembro

Todo sistema padre recuerda para cada hijo h_i, a cuántos saltos RMNP está el miembro más alejado, de entre todos los que están accesibles a través de h_i (HFM_h_i). Esta información es aprendida al recibir paquetes ACK y NAK. Los miembros, al generar un ACK o NAK inicializan el campo HFM a uno, mientras que cada encaminador al generar un ACK o un NAK inicializa el campo HFM de dicho paquete a: (Max HFM_h_i) + 1.

De acuerdo con todo esto, la forma de ajustar MAX_RET_BCR_h_i en un sistema padre es la siguiente:

$$MAX\_RET\_BCR\_h_i = MAX\_RET\_BCR\_FH + HFM\_h_i \qquad (5.4)$$

Siendo, MAX_RET_BCR_FH un parámetro de configuración que indica, cual será el número máximo de retransmisiones a un miembro que esté a un único salto.

Con este procedimiento se consigue que los MAX_RET_BCR_h_i sean más pequeños cuanto más cerca esté el sistema de los miembros, e indirectamente tiene como efecto que las anomalías se detecten en el sistema más cercano al punto donde se originó el problema.

Como consecuencia de las particularidades de la fase de establecimiento: (1) los miembros retardan el proceso de confirmación hasta haber recibido la primera ventana de transmisión, lo cual impide que los sistemas conozcan el número de saltos RMNP al miembro más alejado a través de cada hijo; (2) los SRs no realizan las funciones de almacenamiento y retransmisión, y como consecuencia, no disponen de temporizadores de retransmisión; (3) el árbol RMNP se va construyendo de forma progresiva en esta fase; en el establecimiento, únicamente la fuente es capaz de detectar las anomalías planteadas, llevando cuenta del número de veces que expiran los temporizadores de retransmisión, y siendo el número máximo de retransmisiones el mismo para todos los hijos. Dicho número máximo viene definido por un parámetro de configuración denominado MAX_RET_BCR_C.

5.10.2 Temporizador de fuera de secuencia

El temporizador de fuera de secuencia (de aquí en adelante denominado T_{OS}), es mantenido en los encaminadores y los miembros. Cada vez que uno de estos sistemas recibe un paquete fuera de secuencia, activa T_{OS} y se mantiene a la espera de recibir los paquetes anteriores en la secuencia. Si expira T_{OS} sin haberse recibido dichos paquetes, el sistema correspondiente genera un NAK, que es enviado a su padre.

Como ya se expuso, la principal motivación de T_{OS} es que las redes atravesadas en el camino desde el padre RMNP pueden desordenar, y consiguientemente, la llegada de un paquete fuera de secuencia no implica que el/los anterior(es) se hayan perdido.

Además de la anterior motivación, existe una segunda que será expuesta con un ejemplo. Supóngase que un encaminador a petición de un hijo ha enviado un NAK_*n* y segui-

damente estando dicha solicitud aún pendiente, envía un NAK_(*n-m*), a petición del mismo o de otro hijo distinto. Estos dos NAKs originarán en el sistema retransmisor ascendiente más cercano, sendos procedimientos de retransmisión por solicitud, y lo normal será, que el primer procedimiento de retransmisión (originado por el NAK_*n*) sea abortado con la llegada del NAK_(*n-m*), originándose un segundo procedimiento de retransmisión a partir del paquete *n-m*. Debe considerarse que el mencionado encaminador tras enviar el segundo NAK se mantendrá a la espera de recibir a partir del paquete *n-m*, pero lo normal será que antes de recibir este paquete le lleguen uno o más paquetes fruto del primer procedimiento de retransmisión. El temporizador de fuera de secuencia tiene como efecto colateral, minimizar la probabilidad de que dicho grupo de paquetes, fruto del primer procedimiento de retransmisión, provoquen que el encaminador genere solicitudes de retransmisión innecesarias (el paquete *n-m* y siguientes vienen en camino).

Como puede observarse, el asignar a T_{OS} un valor muy bajo puede originar solicitudes de retransmisión y consecuentemente procesos de retransmisión, innecesarios. Por el contrario, si el valor asignado a T_{OS} es muy alto se tardará mucho tiempo en detectar pérdidas de paquetes, y los sistemas que utilizan este temporizador necesitarán más memoria para almacenar temporalmente, los paquetes recibidos fuera de secuencia. Sin embargo, como puede observarse el valor asignado a T_{OS} no será un factor crítico, de aquí que RMNP no incluya algoritmos específicos que permitan estimar de forma precisa qué valor debe ser asignado a T_{OS}.

Según lo anteriormente expuesto podemos concluir que el administrador de cada sistema (encaminador o miembro), debe analizar los siguientes factores antes de asignar un valor a T_{OS}: (1) si las subredes cercanas al sistema correspondiente desordenan o no, y la frecuencia; (2) estimar la distancia en tiempo al sistema retransmisor (fuente o SR) ascendiente más cercano. Como es obvio el árbol RMNP es dinámico y por consiguiente el administrador no conoce con seguridad cual será el sistema retransmisor ascendiente más cercano. Sin embargo, dicho administrador puede conocer detalles que le permitan estimar un valor adecuado, como por ejemplo, si existen SRs en el mismo dominio, o incluso en la misma subred.

5.10.3 Temporizador de liberación

El temporizador de liberación (de aquí en adelante denominado T_{LIB}), es mantenido en los encaminadores y miembros. T_{LIB} tiene como finalidad evitar que alguno de estos sistemas, por no recibir el correspondiente paquete LIB avisando de la liberación de una cone-

xión o por quedar fuera del árbol RMNP debido a un cambio topológico, no libere los recursos asociados a la mencionada conexión.

Cada vez que un encaminador o un miembro reciben un paquete distribuido por la fuente (un paquete DAT o MTTUD), actualizan T_{LIB} asignándole el valor máximo.

Si en un sistema expira T_{LIB}, éste da por finalizada la conexión y libera los correspondientes recursos locales asociados.

No es aconsejable que T_{LIB} tenga asignado un valor pequeño, lo cual podría causar que erróneamente se diera por finalizada una conexión activa, pero por razones obvias tampoco es aconsejable un valor excesivamente alto. En concreto, para T_{LIB} se recomiendan valores en el rango de 3 a 4 minutos.

5.10.4 Temporizador de estado de la fuente

El temporizador de estado de la fuente (T_{SS}), está relacionado con T_{LIB} y es mantenido en la fuente. T_{SS} tiene como finalidad evitar que la fuente esté un periodo muy largo sin distribuir ningún paquete, hecho que podría provocar que algún sistema erróneamente diera por finalizada la conexión.

Cada vez que la fuente distribuye un paquete (DAT o MTTUD) actualiza T_{SS} asignándole el valor máximo. En el caso de que T_{SS} expire, la fuente envía N_{SS} paquetes DAT vacíos.

Es obvio, que el valor asignado a T_{SS} en la fuente debe estar en consonancia con el asignado a T_{LIB} en los encaminadores y miembros, de aquí que para T_{SS} se recomienden valores en el rango de 2 a 3 minutos.

5.10.5 Temporizador de esperando contestación de mi padre

El temporizador de esperando contestación de mi padre (denominado T_{WF}), es mantenido en los sistemas hijo (miembros y encaminadores). Cada vez que uno de estos sistemas envía un paquete TS informando a su padre de un determinado hecho, por ejemplo de su deseo de entrar a formar parte del árbol RMNP o de su deseo de salir del árbol, activa T_{WF} mientras se mantiene a la espera de recibir la correspondiente confirmación (TS_ACK). En

el caso de que expire T_{WF} el hijo reenvía el paquete TS. Cuando el hijo reciba la mencionada confirmación detendrá el temporizador.

Dado que en condiciones normales, el tiempo que tarda un hijo en recibir el TS_ACK es variable, dependiendo entre otras cosas de la localización del padre y de su carga, para ajustar T_{WF} tras sucesivas retransmisiones, se utiliza una estrategia de incremento exponencial binario, de aquí que:

$$T_{WF} = 2^i T_{WF} \qquad \textbf{(5.5)}$$

Siendo,

i: número de retransmisiones del mismo paquete.

El valor asignado inicialmente a T_{WF} será el indicado por el parámetro T_{WF0}.

Si T_{WF} expira un número máximo de veces permitido, definido por el parámetro MAX_TWF, el comportamiento del sistema dependerá de si este hecho ha ocurrido en un miembro o en un encaminador. Si ocurre en un miembro, éste hace lo siguiente:

- ❒ Indica al nivel superior que se ha producido un aborto de la conexión RMNP así como la causa.
- ❒ Libera los recursos asociados a la comunicación.

Si ocurre en un encaminador, éste hace lo siguiente:

- ❒ Origina una liberación parcial y como consecuencia manda un paquete LIB a sus descendientes, activando el flag AB e indicando la causa. Sólo es preciso enviar un paquete LIB, pues si éste se pierde, sus descendientes detectarán la anomalía al expirar su temporizador T_{LIB}.
- ❒ Libera los recursos asociados a la conexión.

5.10.6 Temporizador de esperando contestación de la fuente

El temporizador de esperando contestación de la fuente (denominado T_{WS}), es mantenido en los miembros y encaminadores. Estos sistemas activan T_{WS} cada vez que envían un paquete a la fuente informando de un determinado hecho: (1) que ha ocurrido un cambio

en el árbol RMNP y como consecuencia debe ponerse en marcha un reinicio; o (2) que el sistema ha originado una liberación parcial (expulsión de uno o más miembros del árbol RMNP), hecho que debe ser notificado al nivel superior. En cualquiera de los casos el sistema emisor, una vez que ha activado T_{WS}, se mantiene a la espera de recibir la correspondiente indicación por parte de la fuente de que ésta recibió el paquete que informaba del evento.

Dado que en condiciones normales, el tiempo que tarda un sistema en recibir la indicación de que la fuente recibió la correspondiente notificación, es variable, dependiendo entre otras cosas de la localización de la fuente respecto al sistema y de su carga, para ajustar T_{WS} tras sucesivas retransmisiones, se utiliza una estrategia de incremento exponencial binario, de aquí que:

$$T_{WS} = 2^i T_{WS} \qquad \textbf{(5.6)}$$

Siendo,

i: número de retransmisiones del mismo paquete.

El valor asignado inicialmente a T_{WS} será el indicado por el parámetro T_{WS0}.

Si T_{WS} expira un número máximo de veces permitido, definido por el parámetro MAX_TWS, el comportamiento del sistema dependerá de si este hecho ha ocurrido en un miembro o en un encaminador. Si ocurre en un miembro, éste hace lo siguiente:

- Indica al nivel superior que se ha producido un aborto de la conexión RMNP así como la causa.
- Libera los recursos asociados a la comunicación.

Si ocurre en un encaminador, éste hace lo siguiente:

- Origina una liberación parcial y como consecuencia manda un paquete LIB a sus descendientes, activando el flag AB e indicando la causa. Sólo es preciso enviar un paquete LIB, pues si éste se pierde, sus descendientes detectarán la anomalía al expirar su temporizador T_{LIB}.
- Libera los recursos asociados a la conexión.

5.11 RMNP versus fiabilidad

Ya se comentó en el capítulo 4, que el hecho de seguir el modelo de grupo de Deering limita el grado de fiabilidad proporcionado. La definición actual del RMNP sigue dicho modelo de grupo, lo cual implica que en determinadas situaciones, que serán poco frecuentes, es posible que un miembro pierda uno o más paquetes. Como ya se ha visto, las medidas adoptadas por el RMNP ante este problema son las siguientes:

1. *Incluir mecanismos que minimicen la probabilidad de pérdidas.* El más importante de estos mecanismos es el utilizado en las fases de establecimiento y reinicio, y consiste en que los miembros retardan la confirmación de paquetes hasta haber recibido la primera ventana completa, en el caso del establecimiento; o todos los paquetes pendientes de confirmación, en el caso del reinicio.

2. *Indicar al nivel superior las situaciones de posible pérdida de paquetes.* En concreto, en RMNP son indicadas al nivel superior las siguientes situaciones:

   - La finalización de la fase de establecimiento.

   - La finalización de cada procedimiento de reinicio.

   - Las liberaciones parciales. Cuando un encaminador expulsa del árbol RMNP a uno o más miembros, éste avisa a la entidad RMNP en la fuente de la liberación parcial y esta entidad indica al nivel superior lo ocurrido.

A continuación se esbozan que mecanismos es necesario incorporar al RMNP para incrementar el grado de fiabilidad proporcionado; anteriormente y con dicho fin, es preciso imponer diversas restricciones que violan el modelo de Deering, y en concreto éstas son las siguientes:

- *La fuente conoce la identidad de los miembros del grupo.* Esto puede conseguirse de dos formas:

  - El nivel superior proporciona explícitamente la lista de los n miembros que desea que reciban todos los paquetes enviados en la conexión RMNP. En este caso, RMNP asume que dichos miembros ya han sido avisados de que deben unirse al grupo, tras lo cual pertenecerán al árbol de distribución. Una pequeña variante consiste en que la fuente proporcione una lista de n miembros, e indique su de-

seo de que al menos k de estos n miembros reciban todos los paquetes enviados en la conexión.

- El nivel superior simplemente indica su deseo, de que todos los paquetes enviados en la conexión sean recibidos por al menos n miembros, sin detallar la identidad de éstos. En este caso, al ser solicitado este servicio, la fuente abre un periodo de suscripción y envía cada cierto tiempo un paquete, avisando del hecho al grupo (mensaje de suscripción abierta). Los miembros, al recibir este mensaje de suscripción abierta, envían a la fuente un mensaje de solicito suscribirme, indicando su identidad y su deseo de recibir todos los paquetes enviados a dicho grupo por dicha fuente. En el momento que n o más miembros, se han suscrito ante la fuente, ésta deja de enviar los mensajes de suscripción abierta y empieza con la fase de establecimiento.

En cualquiera de los dos casos planteados, antes de comenzar el establecimiento de la conexión, la entidad RMNP en la fuente dispone de una lista, con la identidad de los miembros que deben recibir todos los paquetes enviados en la conexión. Dicha lista, independientemente de como se haya obtenido, se denomina *lista de miembros a considerar.*

- *Los miembros no pueden entrar o salir del grupo en cualquier momento.* En concreto, los miembros que deseen recibir los datos enviados durante una conexión RMNP, deben unirse al grupo antes de la fase de establecimiento y deben dejar el grupo posteriormente a la liberación de la conexión. En el caso de que un miembro se una al grupo una vez iniciada la conexión, no se le garantizará que vaya a recibir todos los paquetes enviados al grupo, y en el caso de que un miembro abandone un grupo antes de finalizar la conexión, será borrado de la *lista de miembros a considerar.*

A continuación se analizan los mecanismos que sería preciso introducir en RMNP para incrementar el grado de fiabilidad, asumidas las anteriores restricciones:

- *Mecanismo de registro ante la fuente.* Un miembro, una vez recibido un paquete que avisa de que se ha abierto el periodo de suscripciones, envía a la fuente un paquete TS con el flag de *registro* activado y pone en marcha el temporizador de *esperando contestación de la fuente.* Si expira dicho temporizador sin haber recibido la correspondiente confirmación por parte de la fuente, un TS_ACK con el mencionado flag activado, vuelve a enviar el paquete TS.

❒ *Mecanismo de comprobación de que los miembros incluidos en la lista de miembros a considerar, pertenecen al árbol RMNP.* La fuente al finalizar la fase de establecimiento y cada procedimiento de reinicio, si hay alguno, comprueba que todos los miembros cuya identidad está anotada en la *lista de miembros a considerar* pertenecen al árbol RMNP, hecho que garantiza que recibirán *todos* los paquetes enviados al grupo por la fuente.

Con el fin de facilitar esta comprobación, cada miembro cuando recibe la primera ventana completa, caso de establecimiento, o todos los paquetes pendientes de confirmación, caso de reinicio, además de mandar un ACK a su padre, que se irá agregando por el árbol hasta llegar a la fuente, también envía directamente a la fuente un paquete TS con el flag de *pertenezco al árbol* activado. Seguidamente pone en marcha el temporizador de *esperando contestación de la fuente.* Si expira dicho temporizador sin haber recibido la correspondiente confirmación por parte de la fuente, un TS_ACK con el mencionado flag activado, vuelve a enviar el paquete TS. Cuando la fuente recibe los ACKs que han ido agregándose por el árbol RMNP, y antes de desplazar la ventana, comprueba si algún miembro no le envió el paquete TS indicándole su pertenencia al árbol. En el caso de que algún miembro de los pertenecientes a la *lista de miembros a considerar,* no le haya mandado el paquete TS, vuelve a reenviar la primera ventana, caso del establecimiento, o todos los pendientes de confirmar, caso del reinicio, esta operación se repite un cierto número de veces *n* (parámetro de configuración). Si después de reenviar la ventana *n* veces, todavía hay algún o algunos miembros que no han enviado el mencionado paquete TS, la fuente los borra de la *lista de miembros a considerar* y los anota en la lista de *miembros con problemas.* A continuación desplaza la ventana y continúa enviando nuevos datos.

Como es lógico, en el momento que la *lista de miembros a considerar* tenga menos de *k* identidades (siendo *k* el número mínimo de miembros que debían recibir todos los paquetes, número especificado por el usuario del servicio), la entidad RMNP aborta la conexión avisando al nivel superior de la causa y pasándole la lista de *miembros con problemas.*

❒ *Mecanismo de aprendizaje de miembros descendientes.* Este mecanismo permite que los encaminadores RMNP aprendan la identidad de los miembros descendientes que están accesibles a través de cada uno de sus hijos. Con este fin, los paquetes

ACK incluyen un campo de longitud variable llamado lista de miembros descendientes. El procedimiento seguido es el siguiente:

- Cada vez que un miembro envía un ACK anota su identidad en el campo lista de miembros descendientes.

- Cuando un encaminador recibe un ACK desde un determinado hijo, utilizando la lista de miembros descendientes, aprende qué miembros tiene accesibles a través de dicho hijo. Los encaminadores al realizar el proceso de agregación de ACKs en la lista de miembros descendientes, anotan la identidad de todos los miembros que están accesibles desde cada uno de sus hijos.

Este sencillo procedimiento hace posible que cada encaminador conozca qué miembros tiene accesibles a través de cada uno de sus hijos. Debe observarse, que únicamente es necesario, que los ACKs enviados en las fases de establecimiento y reinicio incluyan esta lista de miembros descendientes, el resto de ACKs llevarán esta lista vacía.

El hecho de que cada encaminador conozca qué miembros están accesibles a través de cada uno de sus hijos, hace posible que cuando un encaminador realice una liberación parcial y expulse del árbol RMNP a todos los descendientes accesibles a través de un determinado hijo, además de avisar a la fuente de la liberación parcial, pueda indicarle la identidad de los miembros que han sido expulsados, esto tiene como finalidad, que éstos sean borrados de la *lista de miembros a considerar* y sean anotados en la lista de *miembros con problemas.*

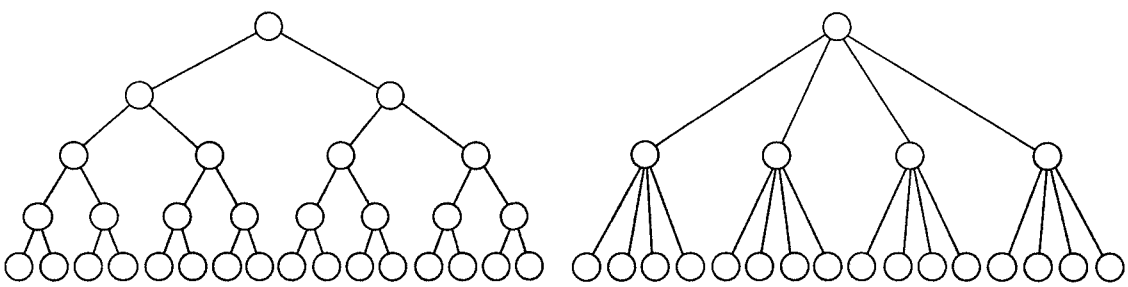
❒ *Mecanismo para avisar a la fuente de la salida de miembros en la fase de transferencia.* Cada vez que un encaminador recibe un paquete TS con el flag ML activado, indicando que un sistema desea dejar de formar parte del árbol RMNP, el encaminador comprueba si este sistema es un miembro (estará en la lista de miembros descendientes), en cuyo caso además de borrarle del árbol, según los procedimientos ya vistos, avisa a la fuente del hecho y le indica la identidad de dicho miembro. Esto permite que la fuente borre a dicho miembro de la *lista de miembros a considerar* y sea anotado en la lista de *miembros con problemas.*

Incorporando los anteriores mecanismos al RMNP, es posible garantizar al usuario del servicio que si la conexión finaliza con una liberación ordenada, todos los miembros, cuya identidad está en la *lista de miembros a considerar,* han recibido todos los paquetes enviados durante la conexión RMNP.

5.12 Número y disposición de los encaminadores RMNP

Los encaminadores RMNP son un recurso caro, de aquí que deba racionalizarse su número y localización. En primer lugar se analiza en general, la problemática del número y localización de los encaminadores RMNP, para pasar a estudiar en particular, el caso de los SRs.

Todos los encaminadores RMNP realizan las funciones de agregación y los distintos tipos de filtrado; hecho que tiene como consecuencia que dichos encaminadores: (1) permitan una optimización del uso de los recursos de red; y (2) eviten problemas de implosión en la fuente.



a) Árbol con 4 niveles y 2 hijos por nodo

b) Árbol con 2 niveles y 4 hijos por nodo

Figura 5.20 Árboles RMNP con distinto número de niveles e hijos por nodo

A la hora de determinar el número y localización de los encaminadores RMNP debe tenerse en consideración que, dado un determinado número de miembros, se consigue la misma reducción de tráfico (gracias a los procesos de agregación y filtrado), disponiendo de un árbol RMNP con n niveles y m hijos por nodo, que con un árbol de m niveles y n hijos por nodo (ver Figura 5.20). Como puede observarse, el optar por una configuración que permita que el árbol RMNP tenga menos niveles y más hijos por nodo, es más barato y por lo tanto más adecuado. Como es obvio, esta solución es más barata por disponer de un menor número de encaminadores RMNP.

5.12.1 Encaminadores SR

Los SRs, además de realizar las funciones vistas en general para los encaminadores RMNP, realizan las funciones de almacenamiento y retransmisión, hecho que hace que la selección de qué encaminadores RMNP se comportarán como SRs sea un aspecto relevante. Estas funciones de almacenamiento y retransmisión hacen que disminuya el retardo medio de distribución y la sobrecarga introducida en la red (ancho de banda y proceso), para obtener una distribución fiable. De aquí que será conveniente sustituir BRs por SRs en las siguientes situaciones:

- ❒ En áreas densas, y esto significa, áreas donde exista una gran concentración de miembros.
- ❒ Cuando el retardo medio de distribución sea un factor crítico.
- ❒ A la entrada de una red con alta probabilidad de errores.

Un encaminador RMNP puede comportarse como un BR o un SR, con la única diferencia de que si se comporta como un SR añadirá a la funcionalidad de un BR las funciones de almacenamiento y retransmisión. La elección de qué encaminadores RMNP deben comportarse como SRs, puede ser estática o dinámica.

En el caso de que la elección sea estática, el administrador de cada red decide qué encaminadores RMNP deben comportarse como SRs y los configura para tal propósito. Por ejemplo, sería deseable que cada encaminador de entrada a un dominio fuera configurado para comportarse como un SR.

Una segunda posibilidad, más atractiva, es hacer que cada encaminador RMNP decida para cada conexión, en función de los recursos que tiene disponibles y de una determinada heurística, si se comporta como un SR o no. La definición actual del RMNP no contempla esta segunda posibilidad.

5.13 RMNP y distintas arquitecturas de encaminamiento multipunto

RMNP puede usarse sobre distintas arquitecturas de encaminamiento multipunto pero el funcionamiento no será en todos los casos igual de óptimo. Al analizar el funcionamiento del RMNP sobre distintas arquitecturas de encaminamiento multipunto deben tenerse en consideración distintos aspectos:

- *Existencia de factores que pueden provocar reinicios innecesarios.* Como ya se ha visto un reinicio supone un gasto en recursos de red y además de esto, ralentiza el desplazamiento de la ventana general en la fuente, disminuyendo el caudal del protocolo. Consiguientemente, es deseable que únicamente se pongan en marcha los reinicios estrictamente necesarios, como por ejemplo, aquellos originados por la caída de un encaminador. A continuación se estudia si las distintas arquitecturas de encaminamiento provocan reinicios innecesarios:

  - En PIM puede cambiar el árbol de distribución al descubrirse rutas punto a punto más óptimas, aún cuando no haya cambios en la topología que obliguen a una reconfiguración del árbol. Estos cambios en el árbol de distribución pueden originar modificaciones en el árbol RMNP, que darán como resultado procedimientos de reinicio innecesarios.

  - En CBT únicamente cambia el árbol de distribución, cuando hay un cambio en la topología (fallos en enlaces o encaminadores), que obliga a que se reconfigure el árbol; pero sin embargo, no se produce ningún cambio en el árbol de distribución mientras se asegure la conectividad, aunque se produzcan cambios en las rutas punto a punto, apareciendo nuevos caminos más óptimos.

    Aunque no se produzcan cambios en el árbol de distribución al aparecer mejores rutas punto a punto, debe analizarse el comportamiento del CBT cuando la fuente no es miembro del grupo. CBT utiliza árboles compartidos y en el caso de que la fuente no sea miembro del grupo, cuando ésta desea enviar un paquete al grupo, lo que hace es mandarlo como un mensaje punto a punto hacia el "core", y a partir del momento en que dicho paquete alcanza el árbol, empieza a distribuirse por éste. Dado este modo de funcionamiento, puede originarse un reinicio innecesario debido a cambio en la ruta punto a punto desde la fuente hasta el árbol de distribución (ver Figura 5.21). Debe observarse que en este caso aunque

no hay ningún cambio en el árbol de distribución, podría producirse una modificación del árbol RMNP.

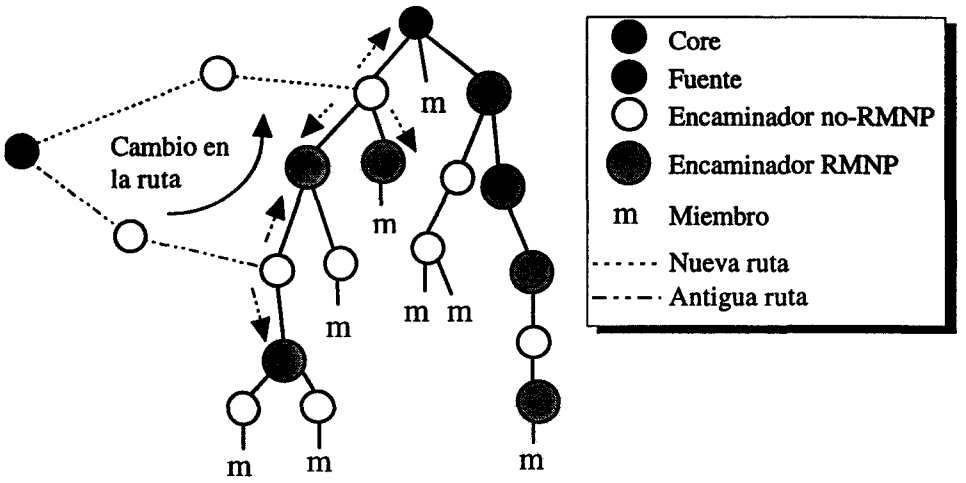


Figura 5.21: Cambio en la ruta punto a punto desde la fuente hasta el *"core"*

- En MOSPF cada vez que entra un nuevo hijo a un grupo, se vuelve a recalcular el árbol de distribución para dicho grupo. Si la reconstrucción del árbol de distribución hace que cambie el árbol RMNP se originará un reinicio claramente innecesario.

❒ *Existencia de bucles.* Otro aspecto a analizar es, si la arquitectura de encaminamiento multipunto garantiza que no existan bucles, debe notarse que la existencia de bucles puede afectar al correcto funcionamiento del RMNP.

❒ *Buena convergencia.* Es deseable que ante cambios en la topología, el algoritmo de encaminamiento multipunto tenga buena convergencia, ya que de no ser así, a nivel RMNP se irán produciendo reinicios sucesivos hasta que el algoritmo converja y el árbol de distribución se mantenga estable.

Capítulo 6.

SIMULACIÓN

6.1 Herramienta de simulación

El MaRS (Maryland Routing Simulator version 2.1) ha sido la herramienta utilizada para simular las principales funcionalidades del RMNP. El MaRS está basado en simulación de sucesos discretos, consecuentemente, el sistema a simular es abstraído por un conjunto de variables de estado y un conjunto de sucesos. El comportamiento del sistema es representado por una secuencia de ocurrencias de sucesos en el tiempo. Cada ocurrencia de un suceso actualiza los valores de las variables de estado y el "tiempo simulado". La principal ventaja de la simulación de sucesos discretos es su gran flexibilidad. Pueden ir introduciéndose de manera progresiva más variables de estado y sucesos, para ir haciendo la simulación más real; obviamente, esto será a costa de un mayor gasto computacional.

Como su nombre indica, el MaRS fue diseñado para simular algoritmos de encaminamiento en redes de área extensa; sin embargo, también permite la simulación de otros protocolos de nivel de red aunque éstos no incluyan la funcionalidad de encaminamiento, como es el caso del RMNP. El MaRS está escrito en C y funciona en entorno UNIX.

La estructura del MaRS es relativamente sencilla, consta de un *motor de simulación* que gestiona la lista de sucesos y el interfaz de usuario, y de una serie de *componentes* que modelan diferentes aspectos del sistema a simular (por ejemplo, un enlace, un nodo, un algoritmo de encaminamiento o una fuente de datos) y ciertas funciones de simulación (por ejemplo, el monitor de rendimiento que se encarga de ir recogiendo estadísticas). Cada componente consiste en una estructura de datos y un conjunto de sucesos cada uno con su

rutina asociada. Una componente es instanciada al inicializar su estructura de datos de forma apropiada. Usualmente para modelar un sistema son precisas múltiples instancias de cada componente. La interconexión de distintas instancias de las componentes permite obtener el modelado de un sistema con topología, protocolo(s) de red y carga de trabajo arbitrarios.

El MaRS permite trabajar con distintos interfaces de usuario; o simplemente, con un interfaz de línea de comandos, o con uno de dos interfaces gráficos opcionales: un interfaz X simple o un interfaz X-Motif. Los dos interfaces gráficos y especialmente el X-Motif son muy atrayentes y permiten mostrar por pantalla la topología de la red, así como la evolución del sistema una vez iniciada la simulación.

Además del aspecto de la flexibilidad ya comentado, otras facilidades que merecen ser reseñadas son que su código fuente está disponible y, su estilo de codificación y documentación es bueno. Consecuentemente, resulta relativamente sencillo realizar modificaciones y hacer extensiones añadiendo nuevas funcionalidades.

El MaRS es un software de libre distribución que se encuentra públicamente disponible vía ftp anonymous en: ftp.cs.umd.edu:/pub/MaRS. Para más detalles sobre el interfaz de usuario y el motor de simulación guiado por sucesos, véase el documento que describe la arquitectura del MaRS [ASDM92] y el manual de usuario [ASDM91].

La principal limitación del MaRS son sus altos requisitos de memoria y sus principales deficiencias están asociadas a la utilización del interfaz X simple. Por ejemplo, con dicho interfaz no es posible la creación o eliminación de instancias de componentes, y resulta muy dificultosa la visualización de redes con un gran número de componentes.

6.2 RMNPsim

En las siguientes subsecciones se presenta en primer lugar que subconjunto de la funcionalidad del RMNP ha sido incorporada al RMNPsim (simulador del RMNP) y posteriormente, cuales son sus principales componentes.

6.2.1 Sistema simulado

A continuación se enumeran cuales de las funciones expuestas en el Capítulo 5 han sido contempladas en el simulador:

- ❒ Distribución básica de paquetes (ver sección 5.4.1).
- ❒ Control intermedio de secuencia (ver sección 5.4.2).
- ❒ Almacenamiento intermedio (ver sección 5.4.3).
- ❒ Agregación de ACKs (ver sección 5.5.1).
- ❒ Generación y filtrado de NAKs (ver secciones 5.6.1 y 5.6.2).
- ❒ Retransmisión (ver sección 5.6.3).

Como puede verse, este subconjunto del RMNP es el núcleo fundamental del protocolo y cubre la mayor parte de su funcionalidad. Adicionalmente, dicho subconjunto se considera suficiente para lograr el principal objetivo de las pruebas de simulación aquí presentadas que consiste en mostrar que el RMNP es un protocolo que escala adecuadamente con relación tanto al número de miembros como al tamaño de la interred.

6.2.2 Componentes

Para desarrollar el RMNPsim ha sido preciso añadir al sistema nuevas componentes, así como modificar algunas de las ya existentes en el MaRS. El resultado han sido 34.000 líneas de código en C, de las cuales aproximadamente 17.500 han sido introducidas para simular el comportamiento específico del RMNP, mientras que el resto han sido reutilizadas del MaRS.

En esta sección se describen las componentes que son simuladas por el RMNPsim y su comportamiento funcional en el simulador:

- ❒ *Encaminamiento multipunto.* Como RMNP es independiente de los protocolos de encaminamiento multipunto, no ha sido necesario simular ninguno de estos protocolos. Tanto el árbol de distribución como el árbol RMNP se construyen al iniciar la simulación y se instalan "estáticamente".

- *Generador de Tráfico Multipunto (Mcast_Source).* Cada instancia de esta componente está conectada a una instancia de una componente fuente. Los paquetes generados son entregados a la correspondiente fuente y van dirigidos a un grupo determinado, identificado por una dirección. Cada instancia de esta componente produce paquetes con un retardo medio entre paquetes distribuido uniforme o exponencialmente.

  Los parámetros de entrada de esta componente incluyen la dirección del grupo destino del tráfico generado, la longitud media de los paquetes de datos, el retardo medio entre paquetes y la distribución seguida.

- *Fuente (Source_Node).* Cada fuente está conectada a un generador de tráfico, del cual recibe los paquetes a distribuir. La fuente puede estar conectada a una componente enlace, o bien a una componente RAL. Cada vez que la fuente recibe un paquete de datos desde el generador de tráfico, ésta envía una copia de dicho paquete a cada uno de sus hijos en el árbol de distribución.

  Los parámetros de entrada de esta componente incluyen entre otros, el valor del temporizador de retransmisión y el tiempo medio de proceso.

- *Enlace y RAL (Link y LAN).* Estas componentes modelan respectivamente un enlace punto a punto y uno multipunto. Cuando un paquete (de datos o de control) es enviado a través de una instancia de una componente RAL, éste es duplicado y cada uno los sistemas conectados a dicha RAL (fuente, encaminador o miembro), recibirá una copia de dicho paquete.

  Los parámetros de entrada de estas componentes incluyen entre otros, el ancho de banda del enlace, el retardo de propagación y la probabilidad de pérdida de paquetes. Entre los parámetros de salida se incluye el número de bytes transmitidos por el enlace.

- *Encaminador (Router_Node).* Esta componente modela el funcionamiento de un encaminador no RMNP. Cada encaminador mantiene información sobre sus hijos y su padre en el árbol de distribución y está conectado a dos o más instancias del tipo enlace o RAL.

Entre los parámetros de entrada de esta componente se incluye el retardo medio en el nodo y entre los parámetros de salida se incluye el número de paquetes recibidos y procesados.

- *Encaminador RMNP (SR_Node).* Esta componente modela el funcionamiento de un encaminador RMNP. Cada encaminador mantiene información sobre sus hijos y su padre, tanto en el árbol de distribución como en el árbol RMNP, y está conectado a dos o más instancias del tipo enlace o RAL.

  Los parámetros de entrada de esta componente incluyen el valor del temporizador de retransmisión, el retardo medio en el nodo. Entre los parámetros de salida incluye el número de paquetes recibidos y procesados, y el número de procedimientos de retransmisión puestos en marcha.

- *Miembro (Member).* Esta componente modela el funcionamiento de un miembro RMNP. Cada instancia miembro esta conectada a una instancia RAL o enlace. Los parámetros de entrada de esta componente incluyen, el tiempo medio de proceso y la probabilidad de pérdida de paquetes.

- *Monitor de Prestaciones (Performance Monitor).* Esta componente recoge estadísticas sobre la red. Las medidas pueden ser actualizadas periódicamente o bien al producirse determinados eventos. Entre la medidas periódicas se incluye la memoria ocupada en la fuente y la memoria media precisada en los SRs, y entre las actualizadas por eventos, el número de paquetes en la red y, el retardo máximo y medio de un paquete.

Con relación al MaRS, el RMNPsim ha mejorado substancialmente su funcionamiento no interactivo. La importancia de este hecho radica en que su uso no interactivo permite aumentar la velocidad del simulador en una relación de uno a diez frente a su utilización interactiva. Dichas mejoras incluyen la posibilidad de especificar mediante un parámetro de entrada el número de paquetes de datos que deben ser enviados por la fuente antes de detenerse la simulación; y adicionalmente permite obtener una imagen de como progresa una simulación en curso enviando al proceso correspondiente una señal de usuario.

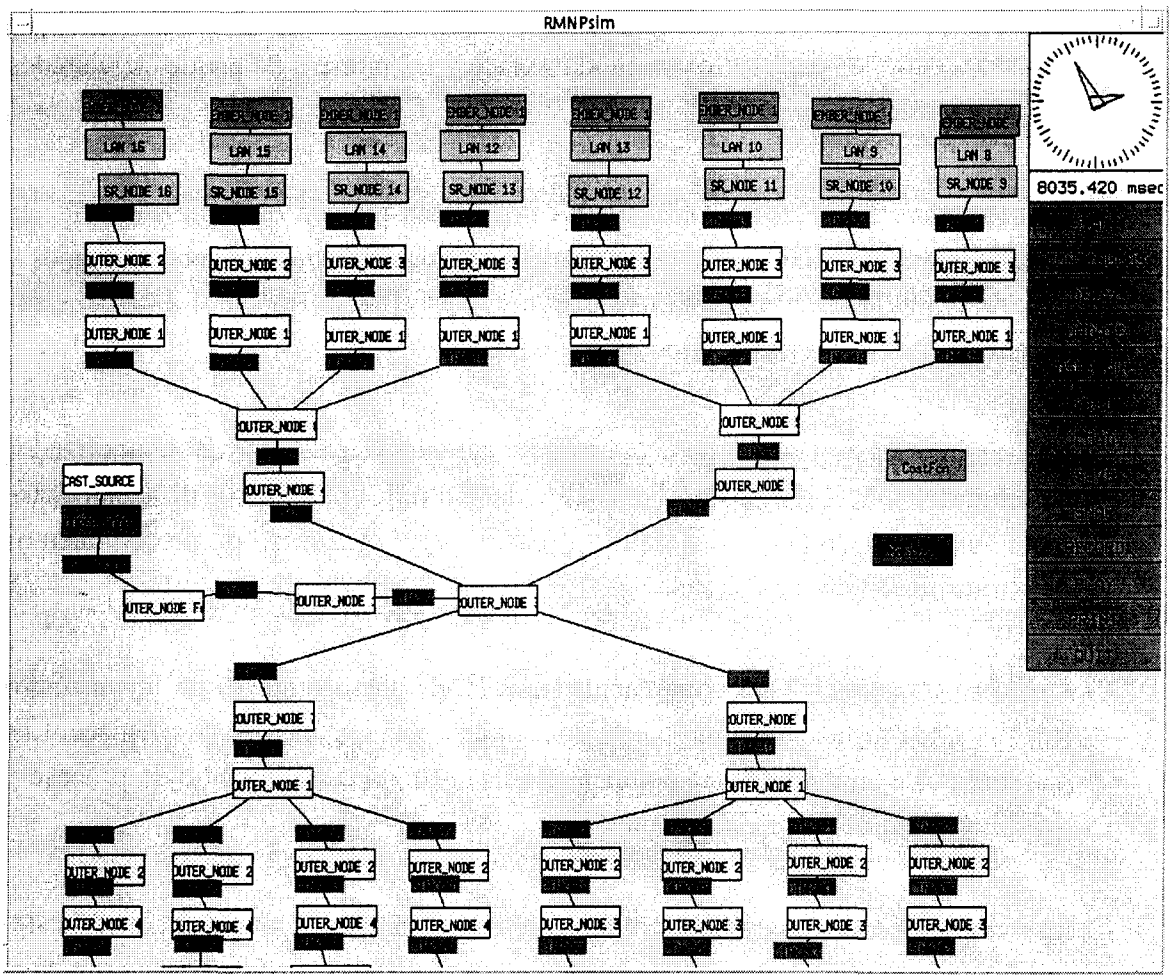


Figura 6.1: Interfaz X del RMNPsim

Además de utilizar el RMNPsim de forma no interactiva también es posible su uso mediante el interfaz X (ver Figura 6.1) o Motif. La Figura 6.2 presenta la apariencia del interfaz Motif, y en concreto se muestra una de las redes utilizadas en las pruebas de simulación realizadas, en la cual cada una de las RALs tiene conectados 25 miembros. Dicho interfaz permite visualizar en cualquier instante los parámetros asociados a un determinado componente, y permite modificar el valor de los parámetros de entrada.

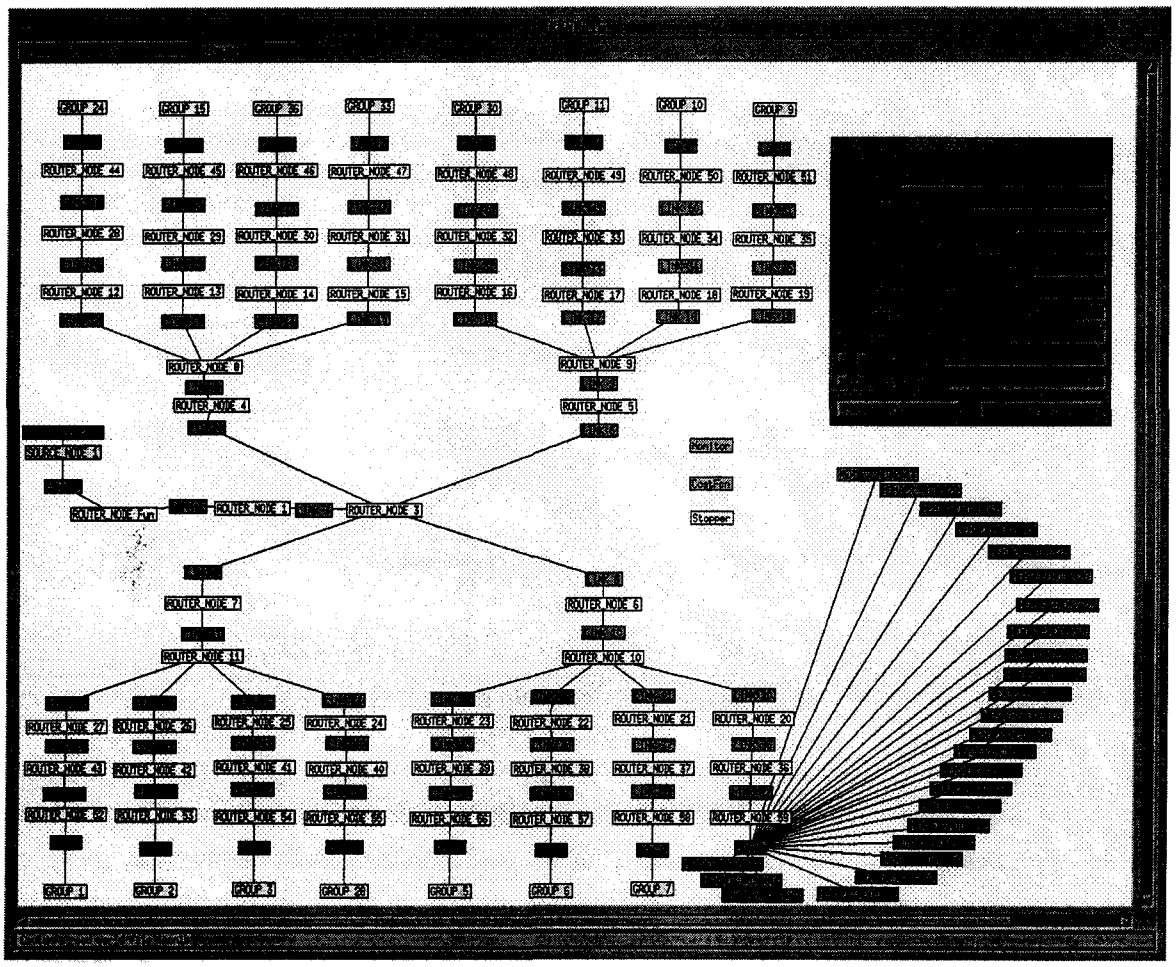


Figura 6.2: Interfaz Motif del RMNPsim

6.3 Topología de red

La Figura 6.3 muestra el modelo de interred que ha sido elegido para realizar las simulaciones: un conjunto de *redes de campus* remotas interconectadas por una *WAN.*

La red WAN está formada por un conjunto de líneas de largo alcance, con un ancho de banda no muy elevado, y encaminadores. La comunicación en dicho tipo de redes se caracteriza por ser relativamente lenta y, propensa a pérdidas y retardos causados principalmente por problemas de congestión. La topología concreta elegida para la WAN (ver Figura 6.4) es un árbol que tiene como raíz la red de campus donde está la fuente, y como hojas, dieciséis redes de campus dónde están distribuidos los miembros del grupo. Con el fin de simplificar la simulación todas las pruebas realizadas se han hecho utilizando la misma topología de WAN (denominada *WAN básica*); sin embargo, como se verá en la sec-

ción 6.6.1, la variación de los distintos parámetros asociados a la red permite modelar redes WAN de distintas dimensiones.

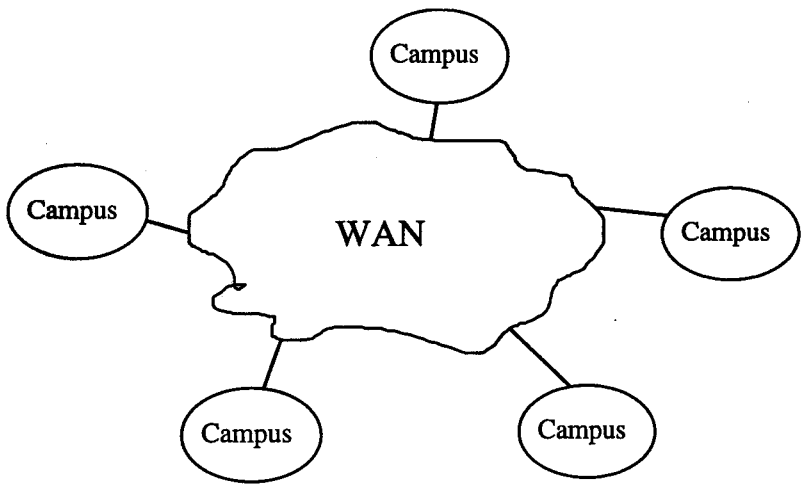
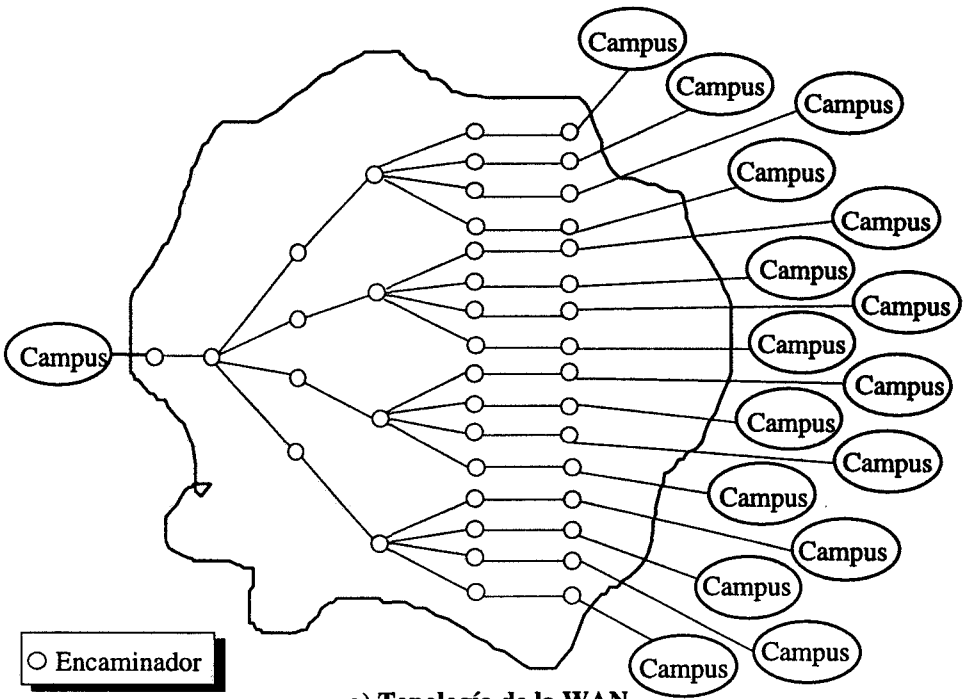


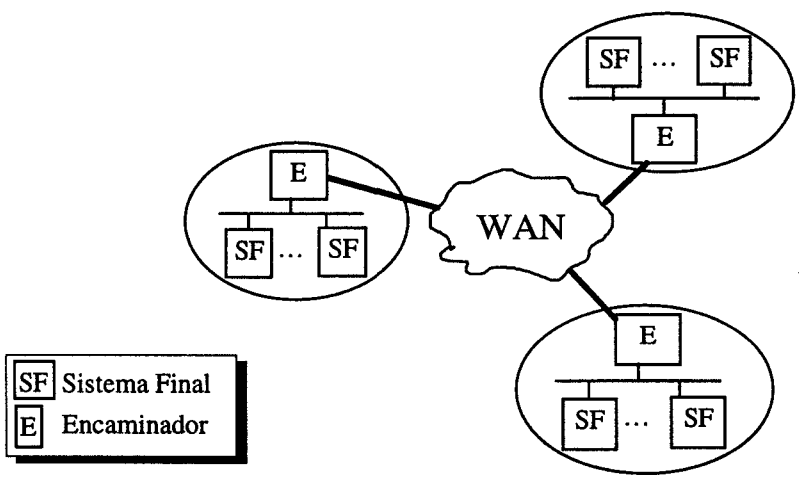
Figura 6.3: Modelo de red

Es preciso reseñar que si la *WAN básica* elegida hubiera tenido más ramificaciones se habrían apreciado mejor las ventajas del RMNP, sin embargo, dado el enorme gasto en CPU que esto hubiera ocasionado, ha sido preciso utilizar un árbol relativamente reducido.

Las redes de campus se caracterizan por estar compuestas por una o más RALs, interconectadas por repetidores, puentes o encaminadores, que proporcionan una comunicación rápida y fiable, siendo alto el ancho de banda disponible. La topología concreta elegida para las redes de campus, se muestra en la Figura 6.4 y consta de una RAL a la que se conectan los distintos sistemas finales (fuente, miembros) y un encaminador que permite la conexión de dicha red a la WAN. Aunque con el fin de simplificar la simulación, las redes de campus estén formadas por una única RAL, los valores asignados a los parámetros de dicha red (retardo de propagación y probabilidad de pérdida) permiten modelar, de forma aproximada, el comportamiento de una red de campus más compleja formada por varias RALs interconectadas por distintos dispositivos (repetidores, puentes y encaminadores).



a) Topología de la WAN



b) Topología de las redes de Campus

Figura 6.4: Topología de la WAN y de las redes de campus

6.3.1 Configuraciones elegidas

Con el fin de probar la bondad del RMNP y de obtener valores de comparación, todas las pruebas de simulación se han realizado sobre tres configuraciones distintas. Las tres configuraciones tienen idénticas topologías de WAN y de redes de campus, mientras que

se ha ido variando el número de encaminadores RMNP, y en concreto el número de SRs. Las características de las configuraciones elegidas son las siguientes:

- **Configuración 1** (ver Figura 6.5). En esta configuración no existe ningún SR, consecuentemente los únicos sistemas RMNP son los sistemas finales (miembros y fuente); los resultados obtenidos en esta configuración pueden ser considerados como los propios de un típico protocolo de multipunto fiable que opera *a nivel de transporte y sigue una aproximación orientada al emisor.*

- **Configuración 2** (ver Figura 6.6). En esta configuración se han introducido SRs en las redes de campus donde están los miembros; esta configuración permite valorar los beneficios obtenidos de realizar las *funciones de agregación.*

- **Configuración 3** (ver Figura 6.7). En esta configuración además de disponer de SRs en las redes de campus, se han introducido algunos de éstos en la WAN; esta configuración permite hacer una valoración de los beneficios obtenidos al disponer de un esquema de *retransmisiones distribuido y con ámbito restringido.*

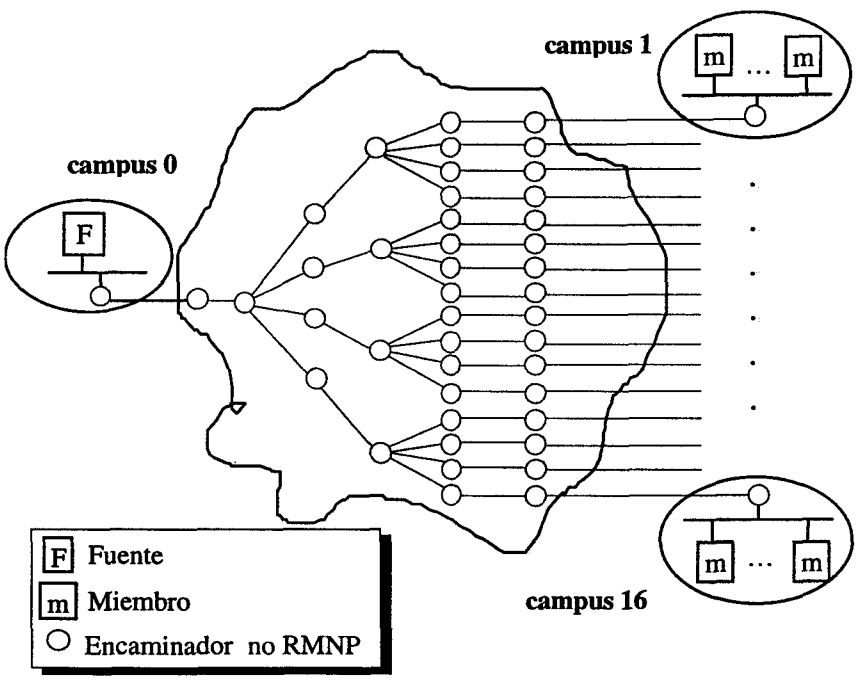


Figura 6.5: Configuración 1

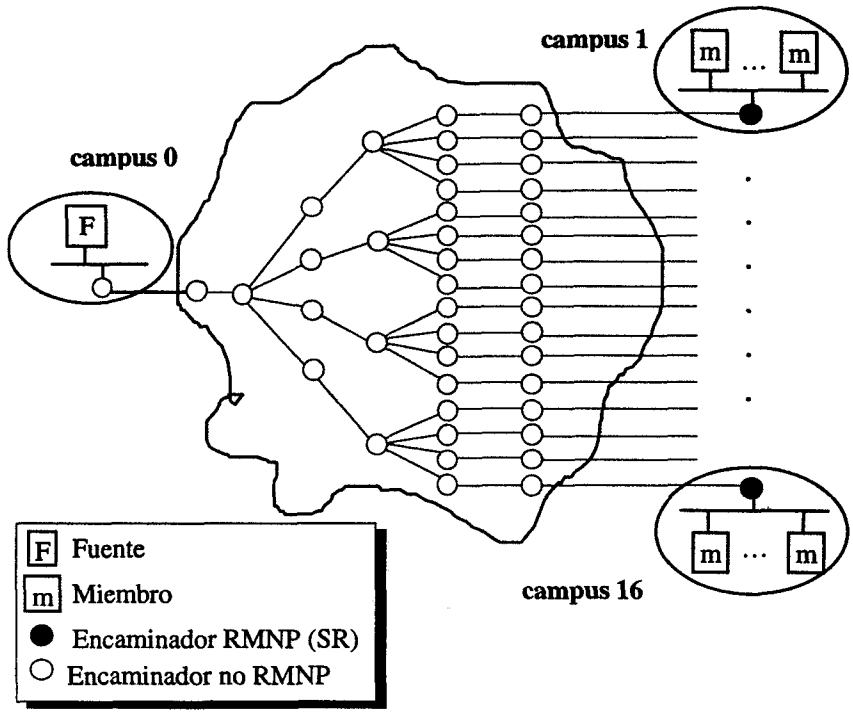


Figura 6.6: Configuración 2

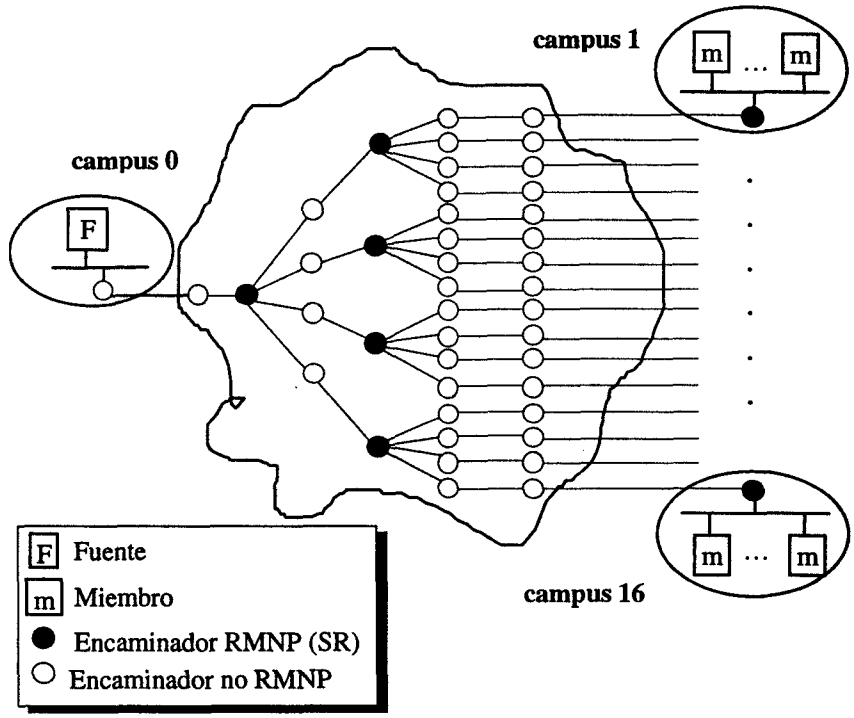


Figura 6.7: Configuración 3

6.4 Parámetros

La Tabla 6.1 muestra los parámetros considerados, así como los valores elegidos para cada uno de éstos.

Parámetro	Valor
Retardo de propagación en un enlace WAN	12 ms
Retardo de propagación en un enlace de campus	5 μs
Ancho de banda en un enlace WAN	2 Mbps
Ancho de banda en un enlace de campus	100 Mbps
Retardo en los encaminadores	6 ms
Tamaño de la ventana de transmisión	10
Distribución seguida para la generación de paquetes	Exponencial
Tráfico generado en el intervalo de simulación	10 Mbytes
Temporizadores de retransmisión	6 IRTT[1]
Longitud de los paquetes de datos	1088 bytes (64 control + 1024 datos)
Longitud de los paquetes ACK	44 bytes
Longitud de los paquetes NAK	44 bytes
Número de miembros	[16, 800]
Probabilidad de pérdida de paquetes en un miembro	0,01 %
Probabilidad de pérdida de paquetes en un salto WAN	0,5 %
Probabilidad de pérdida de paquetes en un salto de campus	0,01 %

Tabla 6.1: Parámetros de simulación

Con el fin de elegir un valor para el parámetro *probabilidad de pérdida de paquetes en un salto WAN*, se han medido los porcentajes de pérdidas en conexiones, a través de la Internet, con sistemas finales situados en distintas partes del mundo (ver Tabla 6.2); todas estas conexiones tienen como origen el mismo sistema final: *asterix.fi.upm.es*, máquina ubicada en la Facultad de Informática de la UPM. Las mediciones se han realizado en tres días distintos y a diferentes horas.

Para medir el porcentaje de pérdidas hasta cada uno de los destinos se ha utilizado el siguiente comando: "*ping -c 500 -i 2 -s 1024 máquina*"; consecuentemente, el sistema origen envía 500 paquetes de 1024 bytes de datos, espaciando en dos segundos el envío de paquetes consecutivos. Para estimar el número de saltos, entre *asterix.fi.upm.es* y cada uno de los destinos, se ha utilizado el comando "*traceroute -m 40*". En cada caso, a partir del porcentaje de pérdidas obtenido (probabilidad de pérdida en el camino completo) y del número de saltos entre el origen y el destino, se ha calculado la probabilidad de pérdida

[1] *Tiempo de ida y vuelta por un interfaz.*

por salto según la expresión (6.2). Obviamente, estos valores son únicamente aproximaciones pero se consideran válidos para estimar un valor para el mencionado parámetro.

Máquina destino	% pérdidas (Ping)			nº saltos (Traceroute)			Probabilidad de pérdida en un salto		
	M1	M2	M3	M1	M2	M3	M1	M2	M3
tenet.berkeley.edu	16	64	16	22	23	22	$3{,}9\ 10^{-3}$	$22\ 10^{-3}$	$4{,}0\ 10^{-3}$
cs.bu.edu	15	34	28	22	20	20	$3{,}6\ 10^{-3}$	$10{,}3\ 10^{-3}$	$8{,}2\ 10^{-3}$
ftp.cs.princeton.edu	21	34	32	20	20	20	$5{,}8\ 10^{-3}$	$10{,}3\ 10^{-3}$	$9{,}6\ 10^{-3}$
e.ergo.cs.cmu.edu	19	35	29	19	20	20	$5{,}5\ 10^{-3}$	$10{,}7\ 10^{-3}$	$8{,}5\ 10^{-3}$
ftp.diku.dk	26	29	29	16	16	16	$9{,}3\ 10^{-3}$	$10{,}6\ 10^{-3}$	$10{,}6\ 10^{-3}$
ftp.cs.city.ac.uk	11	0	15	17	16	16	$3{,}4\ 10^{-3}$	0	$5{,}0\ 10^{-3}$
nic.funet.fi	35	20	31	12	12	13	$17{,}7\ 10^{-3}$	$9{,}2\ 10^{-3}$	$14{,}1\ 10^{-3}$
opera.chorus.fr	29	20	72	18	18	18	$9{,}4\ 10^{-3}$	$6{,}2\ 10^{-3}$	$34{,}7\ 10^{-3}$
duck.df.ki.uni-sb.de	36	39	17	19	19	19	$11{,}6\ 10^{-3}$	$12{,}9\ 10^{-3}$	$4{,}9\ 10^{-3}$
cs.sfu.ca	34	34	16	21	22	21	$9{,}8\ 10^{-3}$	$9{,}4\ 10^{-3}$	$4{,}1\ 10^{-3}$
ftp.mpce.mq.edu.au	37	27	12	23	22	22	$9{,}9\ 10^{-3}$	$7{,}1\ 10^{-3}$	$2{,}9\ 10^{-3}$
dcssoft.anu.edu.au	26	39	10	24	24	24	$6{,}2\ 10^{-3}$	$10{,}2\ 10^{-3}$	$2{,}1\ 10^{-3}$
srawgw.sra.co.jp	39	35	25	22	23	22	$11{,}1\ 10^{-3}$	$9{,}3\ 10^{-3}$	$6{,}5\ 10^{-3}$
toklab.ics.osaka-u.ac.jp	47	34	30	27	26	28	$18{,}4\ 10^{-3}$	$8{,}0\ 10^{-3}$	$6{,}3\ 10^{-3}$

M1 = Medición realizada el viernes 7 Junio de 1996 de 8:00 a 14:36 horas
M2 = Medición realizada el lunes 10 Junio de 1996 de 12:00 a 17:15 horas
M3 = Medición realizada el miércoles 19 Junio de 1996 de 16:00 a 21:46 horas.

Tabla 6.2: Valores experimentales obtenidos con los comandos Ping y Traceroute

A continuación, se deduce la expresión que permite calcular la probabilidad de pérdida en un salto a partir de la probabilidad de pérdida en un camino.

Sea,

p: probabilidad de que un paquete no se pierda en un salto,

$\overline{p}$: probabilidad de que un paquete se pierda en un salto,

P_c: Probabilidad de que un paquete no se pierda en el camino entre dos puntos,

$\overline{P}_c$: Probabilidad de que un paquete se pierda en el camino entre dos puntos,

n: número de saltos.

La expresión de $\overline{P}_c$ es:

$$P_c = p^n = (1-\overline{p})^n$$

$$\overline{P}_c = 1 - P_c = 1 - (1-\overline{p})^n$$

$$\overline{P}_c = 1 - (1-\overline{p})^n \qquad \textbf{(6.1)}$$

despejando $\overline{p}$, se tiene:

$$1-\overline{P}_c=(1-\overline{p})^n$$

$$\sqrt[n]{1-\overline{P}_c}=(1-\overline{p})$$

$$\overline{p}=1-\sqrt[n]{1-\overline{P}_c} \qquad (6.2)$$

Con respecto al valor asignado a los temporizadores de retransmisión debe comentarse que el objetivo perseguido ha sido evitar que se produjeran vencimientos prematuros del temporizador y de aquí que se les haya asignado un valor de 6*IRTT. Sin embargo, este valor no es decisivo para las pruebas de simulación pues como ya se ha comentado, en el RMNP la gran mayoría de las retransmisiones son por solicitud explícita (NAK) en lugar de ser originadas por el vencimiento de un temporizador.

6.5 Métricas

Para evaluar el rendimiento del protocolo se han elegido por su representatividad las siguientes medidas: *el retardo de distribución, la carga introducida en la WAN* y *la memoria precisada en los encaminadores RMNP (SRs).*

Desde el punto de vista de la aplicación y de los usuarios, resulta fundamental que el **retardo de distribución** sea mínimo, y esto significa que los datos enviados por la fuente lleguen lo antes posible a todos los miembros. Para analizar la bondad del RMNP desde esta perspectiva se ha definido el *Retardo Medio de Distribución (R.M.D.)* como, el tiempo medio transcurrido desde que un paquete es liberado por la fuente hasta que es recibido correctamente por todos los miembros del grupo.

$$R.M.D.=\frac{\sum_{1}^{m} r_{p_i}}{m} \qquad (6.3)$$

Siendo,

m: número de paquetes de datos enviados por la fuente,

r_{p_i}: tiempo que se ha tardado en distribuir el paquete p_i a *todos* los miembros del grupo.

El R.M.D. también podría haber sido definido como una media de medias (ver expresión 6.4) e igualmente hubiera sido un estimador válido. Sin embargo, debe recordarse que se planteó como un objetivo de diseño del RMNP que éste soportara las aplicaciones sensibles al retardo, y existen ciertas aplicaciones que utilizan algoritmos distribuidos e

imponen la restricción de que dado un intercambio de resultados parciales los procesos en los distintos sistemas no pueden proseguir hasta que dichos procesos participantes (miembros) no han obtenido el resto de resultados parciales. De aquí que se haya adoptado una postura conservadora y se haya considerado el caso peor, es decir, medir el tiempo desde que un paquetes es liberado por la fuente hasta que es recibido por *todos* los miembros del grupo.

$$R.M.D. = \frac{\sum_{i=1,j=1}^{i=n,j=m} r_{p_i,m_j}}{n*m} \quad (6.4)$$

Siendo,

n: número de paquetes de datos enviados por la fuente,

m: número de miembros,

r_{p_i,m_j}: retardo del paquete p_i en llegar al miembro m_j.

Desde la perspectiva de la red, la bondad del protocolo está directamente relacionada con la carga introducida. Aunque el protocolo proporcione un bajo retardo de distribución si consume muchos recursos, con el riesgo de congestionar la interred, dicho protocolo no tendrá un buen comportamiento. Esto es especialmente importante en las WANs donde los recursos son más escasos y deben ser compartidos por un gran número de usuarios, hechos que causan que el coste asociado a la utilización de recursos WAN sea elevado. La carga introducida en la WAN tiene dos aspectos diferenciados: la carga de proceso en los encaminadores y el consumo de ancho de banda en las líneas.

Para medir la **carga de proceso,** se ha definido la *Carga por Encaminador (C.E.)* como, el número medio de paquetes recibidos y procesados por cada encaminador (RMNP y no-RMNP) perteneciente al árbol de distribución, normalizado respecto al número de paquetes de datos enviados por la fuente.

$$C.E. = \frac{\sum_{1}^{n} p_i}{n \times m} \quad (6.5)$$

Siendo,

m: número de paquetes de datos enviados por la fuente,

n: número de encaminadores (RMNP y no-RMNP) pertenecientes al árbol de distribución,

p_i: número total de paquetes (datos y control) recibidos y procesados por el encaminador *i*.

Análogamente se ha definido la *Carga por Encaminador RMNP (C.E.R)* como, el número medio de paquetes recibidos y procesados por cada encaminador RMNP perteneciente al árbol de distribución, normalizado respecto al número de paquetes de datos enviados por la fuente. Y, la *Carga por Encaminador no RMNP (C.E.nR)* como, el número medio de paquetes recibidos y procesados por cada encaminador no RMNP perteneciente al árbol de distribución, normalizado respecto al número de paquetes de datos enviados por la fuente

Para cuantificar el **consumo de ancho de banda,** se ha definido la *Carga por Línea de la WAN (C.L.W.)* como, el número medio de bytes transmitidos por cada línea de la WAN perteneciente al árbol de distribución, normalizado respecto al número de octetos de datos enviados por la fuente.

$$C.L.W. = \frac{\sum_{1}^{n} B_i}{n \times m} \quad (6.6)$$

Siendo,

m: número de octetos de datos enviados por la fuente,
n: número de líneas en la WAN pertenecientes al árbol de distribución,
B_i : número total de octetos transmitidos por la línea *i.*

Obviamente, el coste asociado a transmitir un byte por una línea no es fijo sino que depende de las características de la propia línea (por ejemplo de su longitud, de su carga, etc.). Consecuentemente, si se desearan comparar los resultados aquí obtenidos con los de otras pruebas de simulación que contemplan diferentes topologías de WAN, por ejemplo para analizar la adecuación del RMNP a distintas configuraciones o para estudiar en que puntos es más conveniente introducir un encaminador RMNP, sería preciso introducir el factor *coste de la línea.* Típicamente, este factor coste sería función de la distancia o la longitud de la línea.

Al incorporar el factor coste de la línea, la expresión para C.L.W. queda como sigue:

$$C.L.W. = \frac{\sum_{1}^{n} B_i C_i}{n \times m} \quad (6.7)$$

Siendo,

m: número de octetos de datos enviados por la fuente,
n: número de líneas en la WAN pertenecientes al árbol de distribución,
B_i: número total de octetos transmitidos por la línea *i,*
C_i: Coste asociado a la línea *i.*

Como puede observarse, la expresión (6.6) es tan sólo una particularización de la expresión (6.7) para el caso de que el coste asociado a cada enlace sea de uno. En las pruebas de simulación aquí realizadas no se ha hecho ningún tipo de comparación entre diferentes topologías y en todos los casos se ha considerado la misma topología de WAN (ver Figura 6.4), de aquí que se haya supuesto que todos los enlaces tienen asociado un coste de uno.

Otro aspecto fundamental al evaluar el rendimiento del RMNP es la **memoria precisada por los SRs**. Para cuantificar estas necesidades de memoria, se ha definido la *memoria por SR* (M_{SR}) como, el número medio de octetos almacenados en los SRs de la WAN. La expresión de M_{SR} es la siguiente:

$$M_{SR} = \frac{\sum_{1}^{n} M_{SR_i}}{n} \quad (6.8)$$

Siendo,

n: número de SRs existentes en la WAN,

M_{SR_i} : número medio de octetos almacenados en el encaminador SR_i.

Al cuantificar las necesidades de memoria en los SRs se han considerado únicamente los SRs de la WAN, pues dado que en los campus la probabilidad de pérdida de paquetes es relativamente baja y la velocidad de transmisión alta, si lo comparamos con la WAN, los SRs situados en los campus no tendrán muchos requisitos de memoria, sino que los puntos críticos respecto a las necesidades de memoria serán los SRs situados en la WAN.

Adicionalmente, se han definido la *memoria de pico en los SRs* (MP_{SR}), como el número máximo de paquetes de datos almacenados en un SR de la WAN durante una comunicación, y la *memoria media en la fuente* (M_F) como el número medio de paquetes de datos almacenados en la fuente.

6.6 Pruebas de simulación realizadas

Se han realizado dos baterías de pruebas diferentes, en la primera de ellas se ha variado el tamaño de la interred mientras que en la segunda se ha variado el tamaño del grupo.

6.6.1 Variando el tamaño de la Interred

En estas simulaciones se ha mantenido fijo el número de miembros (400, con 25 miembros por campus) mientras que se ha ido variando el tamaño de la interred; en concreto, lo que se ha ido modificando en las distintas pruebas ha sido la dimensión de la WAN. Esta batería de pruebas se ha repetido para dos caudales de entrada de paquetes en la red distintos, en concreto se han considerado dos valores distintos de retardo medio entre paquetes (R_{MP}), siendo éstos de 0,5 sg. y 0,4 sg. Los resultados de estas simulaciones se muestran en las Tablas 6.4 y 6.5 respectivamente.

Como ya se ha comentado, la topología de WAN utilizada en todas las simulaciones se ha mantenido invariable y ha sido la denominada *WAN básica* (ver Figura 6.4). La única motivación de utilizar siempre la misma topología ha sido simplificar las tareas de simulación. Sin embargo, como se analiza a continuación es posible modelar WANs de distintas dimensiones utilizando una única topología.

Para modelar una WAN de mayor dimensión utilizando la *WAN Básica* deben considerarse los siguientes aspectos:

- *Al aumentar el tamaño de la WAN se incrementa el retardo de tránsito de cada paquete*. Esto tiene dos justificaciones, en primer lugar, lo normal será que haya un incremento en el número de encaminadores atravesados; y por otro lado, bien aumentará el número de líneas atravesadas, bien será mayor la longitud de éstas, o ambas cosas a la vez.

- *Al aumentar el tamaño de la WAN se incrementa la probabilidad de que un paquete se pierda en su camino entre el origen y el destino*; esto es debido a que lo normal será que aumente el número de saltos y aunque la probabilidad de pérdida en cada uno de éstos se mantenga constante, se incrementará la probabilidad de que un paquete se pierda y no llegue a su destino.

- *Al aumentar el tamaño de la WAN se incrementa el retardo de tránsito*, consecuentemente para obtener un caudal equivalente es necesario que la ventana de transmisión aumente proporcionalmente con relación al retardo.

Consecuentemente, para modelizar redes WAN de distintas dimensiones utilizando la *WAN básica* se ha hecho lo siguiente:

- Si r es el retardo en los encaminadores de la *WAN básica*, para modelar una WAN_i, λ veces mayor, el retardo en los encaminadores será λr.

- Si el retardo de propagación en un enlace de la *WAN básica* es t_p, para modelar una WAN_i, λ veces mayor, el retardo de propagación considerado será λt_p.

- Si el porcentaje de pérdida en un salto de la *WAN básica* es $\overline{p}$, para modelar una WAN_i, λ veces mayor, el porcentaje de pérdida en un salto de la WAN_i será $\overline{p}$' y vendrá dado por la expresión (6.9). A continuación se expone la deducción de dicha expresión:

  Ya se ha visto en la sección 6.4 que la expresión que relaciona la probabilidad de pérdida en un camino de n saltos ($\overline{P}_c$) con la probabilidad de pérdida en un salto ($\overline{p}$) es:

$$\overline{P}_c = 1-(1-\overline{p})^n$$

  Para modelizar un camino de λn saltos y con probabilidad de pérdida por salto de $\overline{p}$, mediante un camino de únicamente n saltos, la probabilidad de pérdida por salto a considerar ($\overline{p}$ ') debe ser la siguiente:

$$\begin{aligned} 1-(1-\overline{p})^{\lambda n} &= 1-(1-\overline{p}')^n \\ (1-\overline{p})^{\lambda n} &= (1-\overline{p}')^n \\ (1-\overline{p})^{\lambda} &= (1-\overline{p}') \\ \overline{p}' &= 1-(1-\overline{p})^{\lambda} \quad (6.9) \end{aligned}$$

- Si la *WAN básica* dispone de una ventana de transmisión de tamaño W_0, el retardo de tránsito es R_0 y se desea modelar una WAN_i con un retardo de tránsito R_i, para calcular el tamaño de la ventana de transmisión (W_i) de forma que el caudal sea equivalente, se utilizará la siguiente expresión:

$$\text{Dado } k = \frac{R_0}{W_0}, \qquad W_i = \frac{R_i}{k} \qquad (6.10)$$

Siguiendo los anteriores criterios, utilizando la *WAN básica* se han modelado cinco WANs distintas (WAN_1...WAN_5), para ello se han ido variando los siguientes parámetros: probabilidad de pérdida de paquetes en un salto WAN, retardo en los encaminadores, retardo de propagación en un enlace WAN y tamaño de la ventana de transmisión; en la Tabla 6.3 se muestran los valores elegidos.

Parámetro / WAN a modelizar	r	t_p	$\overline{p}$	W_i
WAN_1 (1,5 veces *WAN básica*)	9	18	0,007	14
WAN_2 (2 veces *WAN básica*)	12	24	0,009	19
WAN_3 (2,5 veces *WAN básica*)	15	30	0,012	23
WAN_4 (3 veces *WAN básica*)	18	36	0,015	28
WAN_5 (3,5 veces *WAN básica*)	21	42	0,017	32

r = Retardo en un encaminador (ms) t_p = Retardo de propagación en un enlace WAN (ms)
$\overline{p}$ = Probabilidad de pérdida en un salto WAN W_i = Ventana de transmisión

Tabla 6.3: WANs de distintas dimensiones

Configuración / Tamaño WAN	Configuración1	Configuración2	Configuración3
WAN Básica	R.M.D.= 527,05 C.E. = 88,65 C.E.nR. = 88,65 C.L.W.= 5,25 M_F = 2,35 T_S = 83,49	R.M.D.= 417,09 C.E.= 11,20 C.E.R.= 26,62 C.E.nR. = 5,46 C.L.W.= 1,72 M_F = 1,51 T_S = 82,15	R.M.D.= 246,86 C.E.= 8,81 C.E.R.= 21,07 C.E.nR. = 2,03 C.L.W.= 1,12 M_{SR}= 0,5 MP_{SR}= 7 M_F = 1,53 T_S = 83,87
WAN_1	R.M.D.= 1.249,78 C.E. = 92,26 C.E.nR. = 92,26 C.L.W.= 5,62 M_F = 3,62 T_S = 84,71	R.M.D.= 1.076,14 C.E.= 11,80 C.E.R.= 26,86 C.E.nR. = 6,19 C.L.W.= 1,99 M_F = 2,54 T_S = 83,69	R.M.D.= 386,87 C.E.= 8,83 C.E.R.= 21,09 C.E.nR. = 2,06 C.L.W.= 1,10 M_{SR}= 0,75 MP_{SR}= 14 M_F = 1,70 T_S = 83,67
WAN_2	R.M.D.= 2.432,6 C.E. = 110,96 C.E.nR. = 110,96 C.L.W.= 6,67 M_F = 4,70 T_S = 84,79	R.M.D.= 2.330,47 C.E.= 12,72 C.E.R.= 27,23 C.E.nR. = 7,32 C.L.W.= 2,39 M_F = 4,21 T_S = 85,56	R.M.D.= 662,74 C.E.= 8,85 C.E.R.= 21,10 C.E.nR. = 2,08 C.L.W.= 1,12 M_{SR}= 1,05 MP_{SR}= 18 M_F = 2,40 T_S = 82,33
WAN_3	R.M.D.= 5.011,79 C.E. = 130,55 C.E.nR. = 130,55 C.L.W.= 7,95 M_F = 7,12 T_S = 84,57	R.M.D.= 4.484,48 C.E.= 14,11 C.E.R.= 27,83 C.E.nR. = 9,01 C.L.W.= 3,03 M_F = 6,85 T_S = 84,78	R.M.D.= 1.107,37 C.E.= 8,89 C.E.R.= 21,14 C.E.nR. = 2,12 C.L.W.= 1,15 M_{SR}= 1,45 MP_{SR}= 19 M_F = 3,28 T_S = 82,79
WAN_4	R.M.D.= 10.464,4 C.E. = 160,09 C.E.nR. = 160,09 C.L.W.= 9,89 M_F = 10,82 T_S = 86,99	R.M.D.= 7.933,39 C.E.= 16,21 C.E.R.= 28,68 C.E.nR. = 11,58 C.L.W.= 4,03 M_F = 10,60 T_S = 87,06	R.M.D.= 1.959,05 C.E.= 8,95 C.E.R.= 21,21 C.E.nR. = 2,17 C.L.W.= 1,20 M_{SR}= 1,95 MP_{SR}= 27 M_F = 4,55 T_S = 85,13
WAN_5	R.M.D.= 15.608,7 C.E.= 177,8 C.E.nR. = 177,8 C.L.W.= 11,14 M_F = 13,71 T_S = 86,81	R.M.D.= 11.767,5 C.E.= 18,24 C.E.R.= 29,60 C.E.nR. = 14,01 C.L.W.= 4,97 M_F = 14,52 T_S = 89,76	R.M.D.= 2.635,34 C.E.= 8,99 C.E.R.= 21,25 C.E.nR. = 2,21 C.L.W.= 1,21 M_{SR}= 2,56 MP_{SR}= 32 M_F = 5,90 T_S = 82,07

R.M.D.= Retardo Medio de Distribución (ms)
C.E.= Carga por Encaminador (paquetes)
C.E.R.= Carga por Encaminador RMNP (paquetes)
C.E.nR.= Carga por Encaminador no RMNP (paquetes)
C.L.W.= Carga por Línea de la WAN (bytes)
M_{SR}= Memoria por SR de la WAN (paquetes)
MP_{SR}= Memoria de pico en los SRs (paquetes)
M_F= Nº medio de paquetes almacenados en la fuente (paquetes)
T_S= Tiempo simulado (minutos)

Tabla 6.4: Pruebas realizadas variando el tamaño de la WAN (R_{MP}= 0,5)

Configuración / Tamaño WAN	Configuración1	Configuración2	Configuración3
WAN Básica	R.M.D.= 554,35 C.E. = 88,79 C.E.nR. = 88,79 C.L.W.= 5,27 M_F = 2,74 T_S = 65,67	R.M.D.= 415,39 C.E.= 11,18 C.E.R.= 26,61 C.E.nR. = 5,43 C.L.W.= 1,71 M_F = 1,76 T_S = 66,79	R.M.D.= 231,83 C.E.= 8,81 C.E.R.= 21,07 C.E.nR. = 2,04 C.L.W.= 1,08 M_{SR}= 0,59 MP_{SR}= 9 M_F = 1,33 T_S = 65,84
WAN_1	R.M.D.= 1.339,65 C.E. = 93,42 C.E.nR. = 93,42 C.L.W.= 5,70 M_F = 3,84 T_S = 68,63	R.M.D.= 1.106,52 C.E.= 11,99 C.E.R.= 26,94 C.E.nR. = 6,42 C.L.W.= 2,06 M_F = 3,12 T_S = 68,88	R.M.D.= 395,25 C.E.= 8,83 C.E.R.= 21,09 C.E.nR. = 2,06 C.L.W.= 1,10 M_{SR}= 0,92 MP_{SR}= 13 M_F = 2,07 T_S = 65,29
WAN_2	R.M.D.= 2.718,45 C.E. = 114,72 C.E.nR. = 114,72 C.L.W.= 6,91 M_F = 5,87 T_S = 68,15	R.M.D.= 2.717,15 C.E.= 13,02 C.E.R.= 27,37 C.E.nR. = 7,69 C.L.W.= 2,53 M_F = 5,60 T_S = 66,72	R.M.D.= 646,19 C.E.= 8,85 C.E.R.= 21,10 C.E.nR. = 2,08 C.L.W.= 1,12 M_{SR}= 1,25 MP_{SR}= 19 M_F = 2,83 T_S = 66,76
WAN_3	R.M.D.= 6.463,3 C.E. = 142,22 C.E.nR. = 142,22 C.L.W.= 8,70 M_F = 9,71 T_S = 69,90	R.M.D.= 5.141,57 C.E.= 14,71 C.E.R.= 28,09 C.E.nR. = 9,74 C.L.W.= 3,31 M_F = 8,74 T_S = 72,87	R.M.D.= 1.241,26 C.E.= 8,90 C.E.R.= 21,15 C.E.nR. = 2,13 C.L.W.= 1,16 M_{SR}= 1,81 MP_{SR}= 23 M_F = 4,15 T_S = 66,92
WAN_4	R.M.D.= 12.045,4 C.E. = 175,39 C.E.nR. = 175,39 C.L.W.= 10,88 M_F = 13,89 T_S = 72,22	R.M.D.= 7.557,03 C.E.= 17,09 C.E.R.= 29,04 C.E.nR. = 12,64 C.L.W.= 4,03 M_F = 12,98 T_S = 72,91	R.M.D.= 2.000,94 C.E.= 8,96 C.E.R.= 21,22 C.E.nR. = 2,19 C.L.W.= 1,21 M_{SR}= 2,51 MP_{SR}= 28 M_F = 5,80 T_S = 66,40
WAN_5	R.M.D.= 424.323 C.E.= 462,32 C.E.nR. = 462,32 C.L.W.= 28,01 M_F = 26,62 T_S = 98,91	R.M.D.= 10.996 C.E.= 18,87 C.E.R.= 29,86 C.E.nR. = 14,78 C.L.W.= 5,28 M_F = 17,36 T_S = 77,67	R.M.D.= 3.232,4 C.E.= 9,03 C.E. R.= 21,30 C.E.nR. = 2,25 C.L.W.= 1,25 M_{SR}= 3,41 MP_{SR}= 32 M_F = 7,99 T_S = 66,72

R.M.D.= Retardo Medio de Distribución (ms)
C.E.= Carga por Encaminador (paquetes)
C.E.R.= Carga por Encaminador RMNP (paquetes)
C.E.nR.= Carga por Encaminador no RMNP (paquetes)
C.L.W.= Carga por Línea de la WAN (bytes)
M_{SR}= Memoria por SR de la WAN (paquetes)
MP_{SR}= Memoria de pico en los SRs (paquetes)
M_F= Nº medio de paquetes almacenados en la fuente (paquetes)
T_S= Tiempo simulado (minutos)

Tabla 6.5: Pruebas realizadas variando el tamaño de la WAN (R_{MP}= 0,4)

A continuación se muestran gráficamente algunos de los resultados obtenidos:

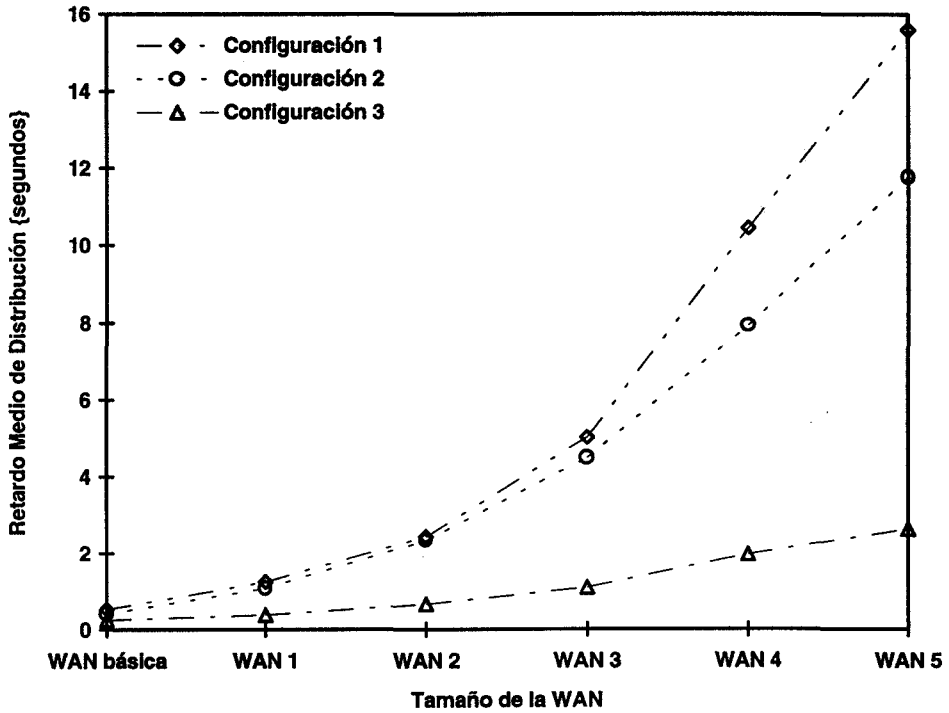


Figura 6.8: R.M.D. versus tamaño de la WAN (R_{MP}= 0,5)

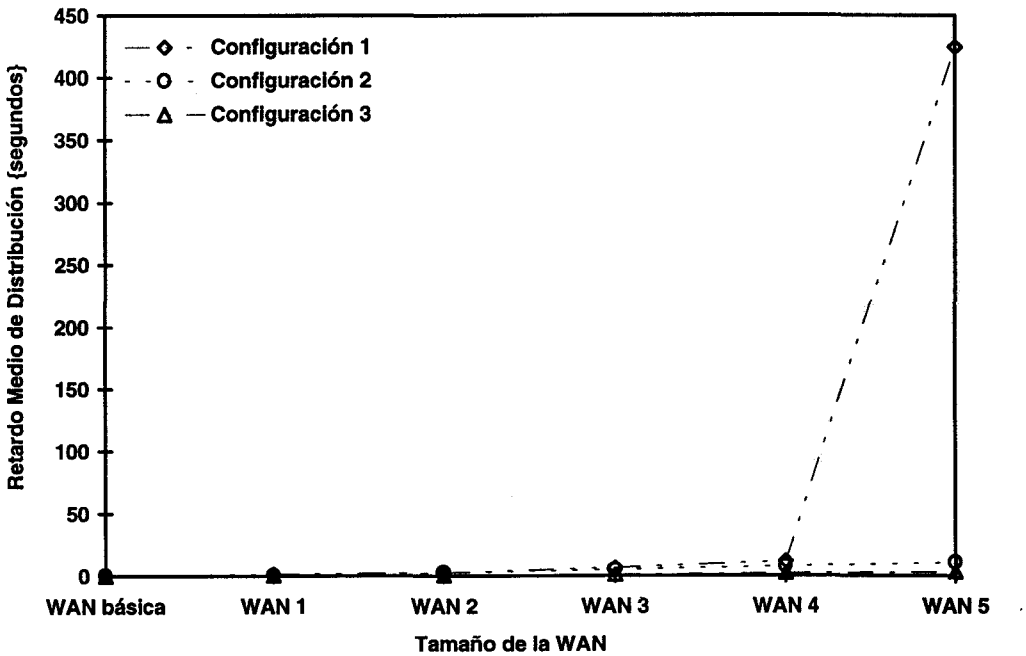


Figura 6.9: R.M.D. versus tamaño de la WAN (R_{MP}= 0,4)

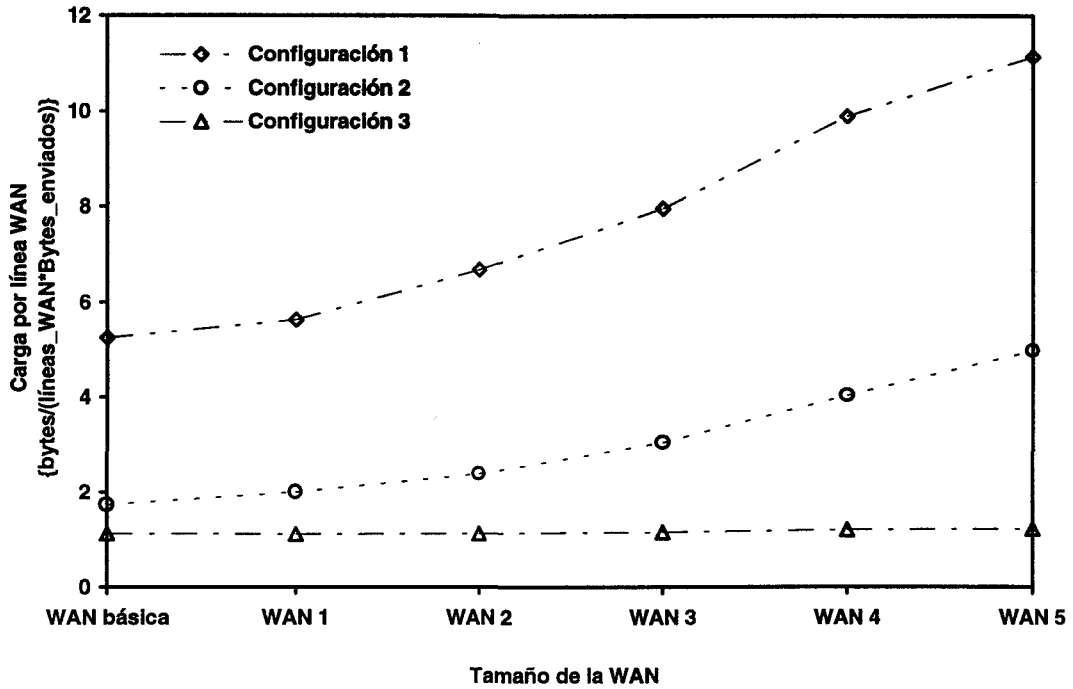


Figura 6.10: C.L.W. versus tamaño de la WAN (R_{MP}= 0,5)

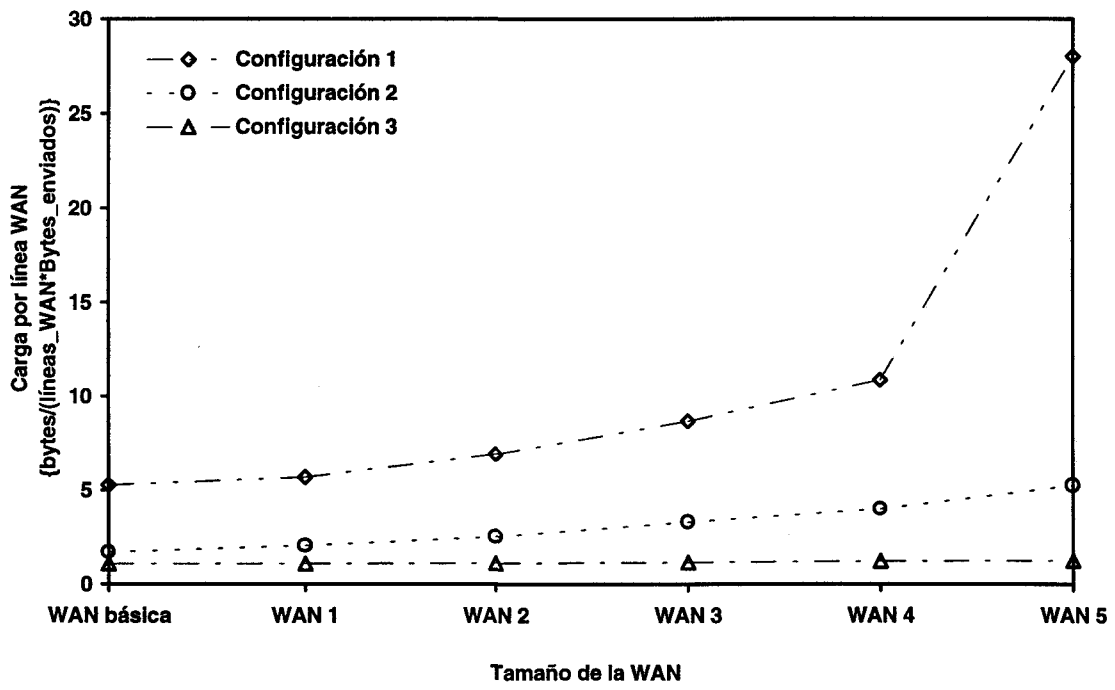


Figura 6.11: C.L.W. versus tamaño de la WAN (R_{MP}= 0,4)

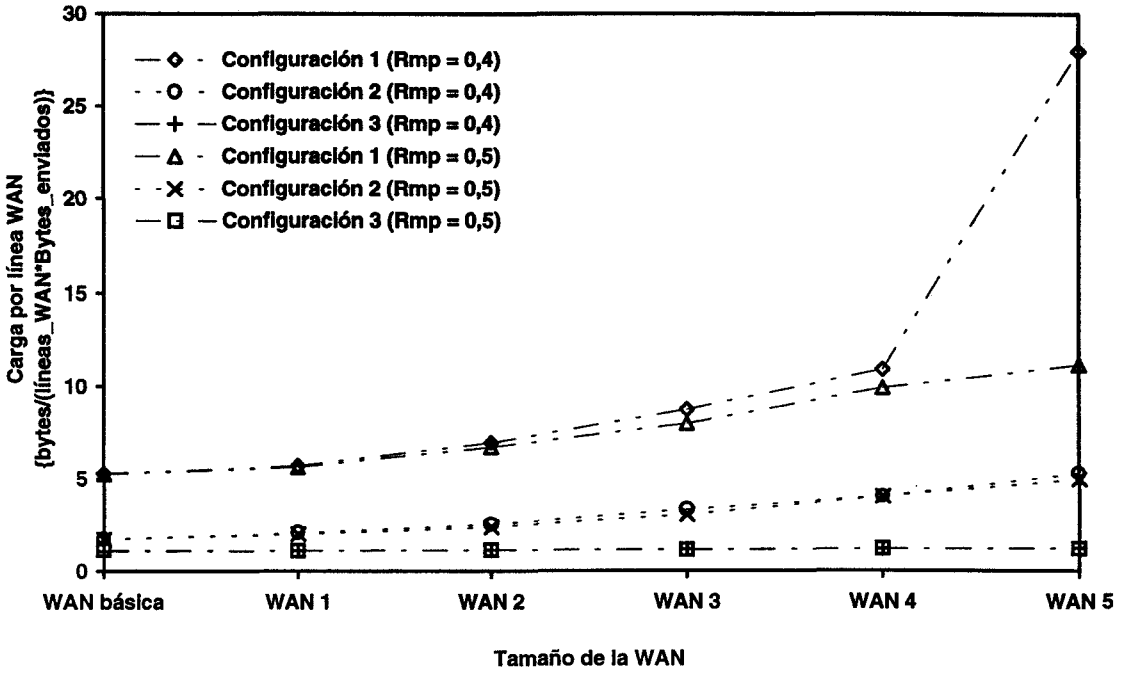


Figura 6.12: C.L.W. versus tamaño de la WAN

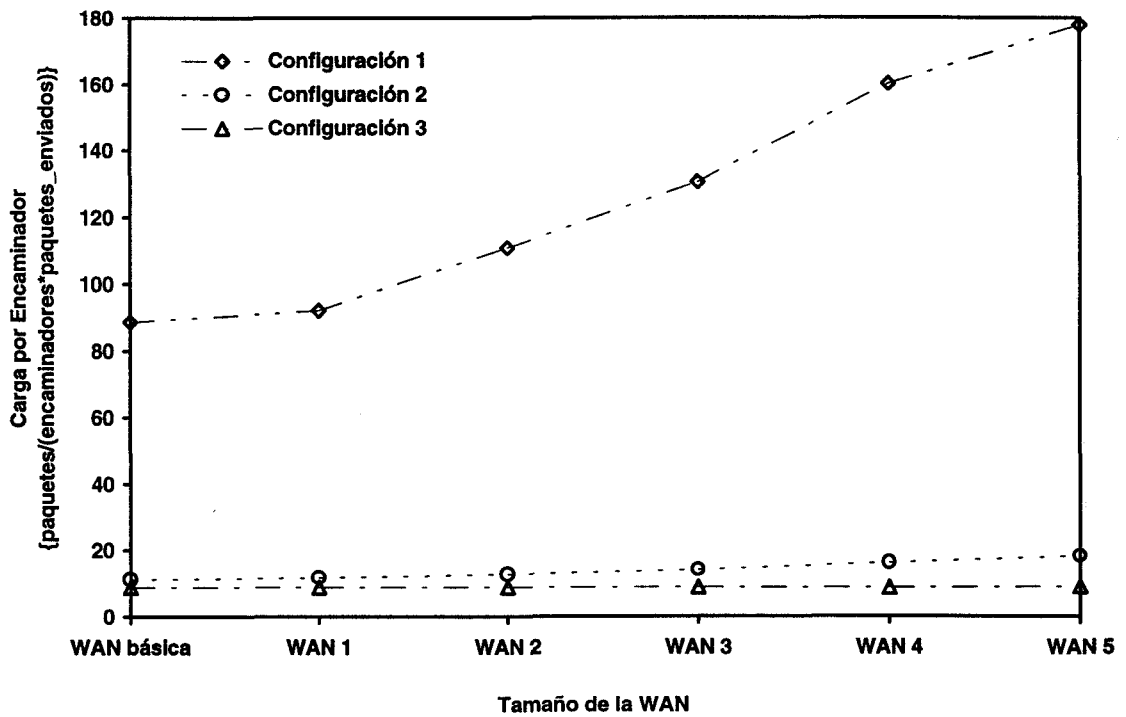


Figura 6.13: C.E. versus tamaño de la WAN (R_{MP}= 0,5)

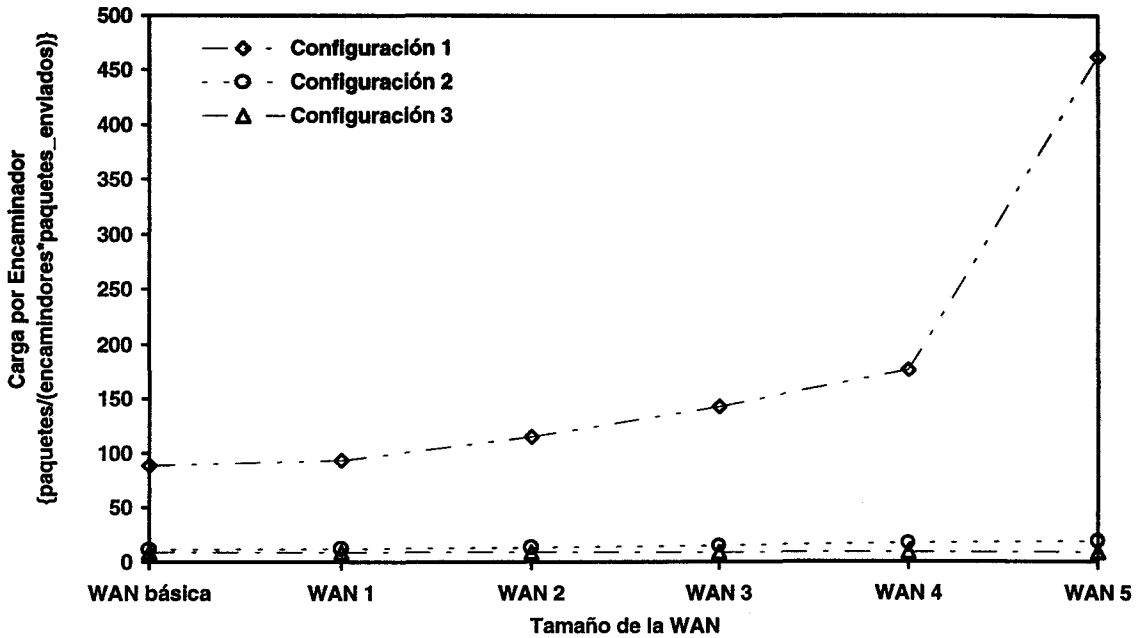


Figura 6.14: C.E. versus tamaño de la WAN (R_{MP}= 0,4)

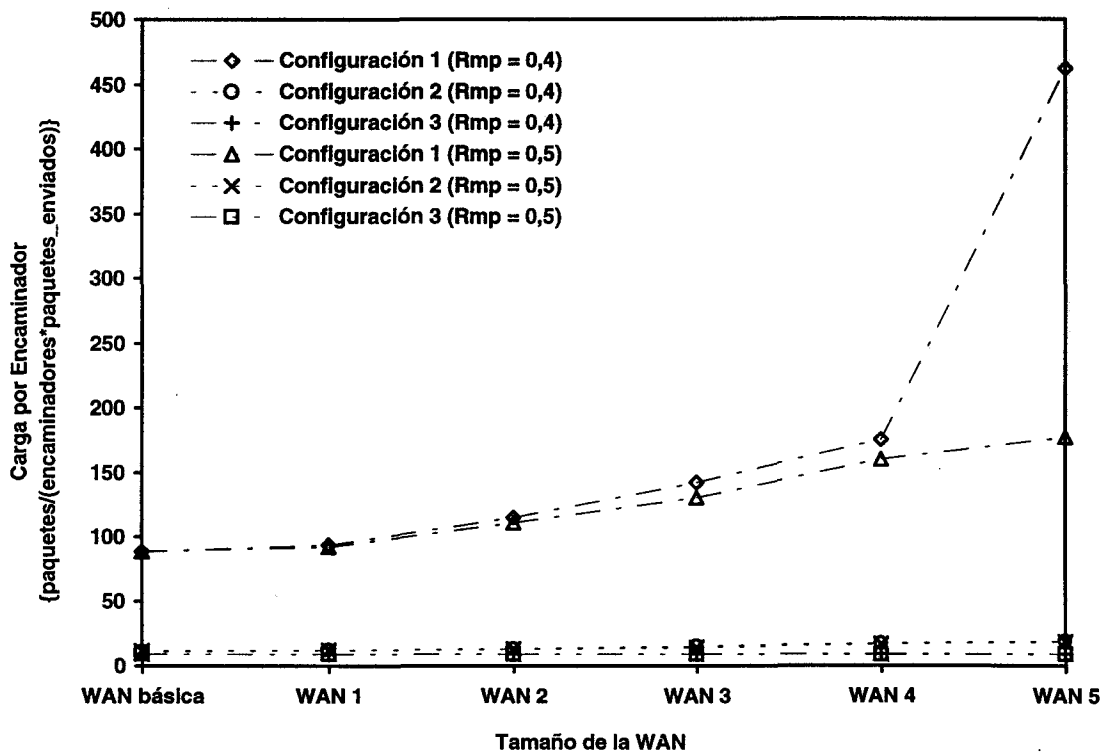


Figura 6.15: C.E. versus tamaño de la WAN

6.6.2 Variando el número de miembros

En esta batería de pruebas se ha utilizado la *WAN básica* y se ha ido variando el número de miembros del grupo. Estas simulaciones se han repetido para dos caudales de entrada de paquetes en la red distintos, en concreto se han considerado dos valores distintos de R_{MP}, siendo éstos de 0,4 sg. y 0,2 sg. Los resultados de estas simulaciones se muestran en las Tablas 6.6 y 6.7 respectivamente.

Configuración / N° miembros	Configuración1	Configuración2	Configuración3
16	R.M.D.= 476,37 C.E. = 4,78 C.E.nR. = 4,78 C.L.W.= 1,70 M_F = 1,64 T_S = 65,95	R.M.D.= 459,57 C.E.= 4,72 C.E.R.= 2,57 C.E.nR. = 5,53 C.L.W.= 1,74 M_F = 1,80 T_S = 66,70	R.M.D.= 227,82 C.E.= 2,28 C.E.R.= 2,73 C.E.nR. = 2,03 C.L.W.= 1,08 M_{SR}= 0,58 MP_{SR}= 8 M_F = 1,31 T_S = 66,45
192	R.M.D.= 516,85 C.E. = 42,17 C.E.nR. = 42,17 C.L.W.= 3,31 M_F = 2,37 T_S = 67,54	R.M.D.= 465,47 C.E.= 7,71 C.E.R.= 13,58 C.E.nR. = 5,52 C.L.W.= 1,74 M_F = 1,81 T_S = 67,45	R.M.D.= 239,36 C.E.= 5,27 C.E.R.= 11,13 C.E.nR. = 2,04 C.L.W.= 1,08 M_{SR}= 0,58 MP_{SR}= 10 M_F = 1,32 T_S = 66,71
400	R.M.D.= 579,13 C.E. = 89,27 C.E.nR. = 89,27 C.L.W.= 5,30 M_F = 2,73 T_S = 67,01	R.M.D.= 457,40 C.E.= 11,27 C.E.R.= 26,65 C.E.nR. = 5,55 C.L.W.= 1,75 M_F = 1,83 T_S = 66,36	R.M.D.= 235,56 C.E.= 8,81 C.E.R.= 21,07 C.E.nR. = 2,03 C.L.W.= 1,08 M_{SR}= 0,58 MP_{SR}= 10 M_F = 1,32 T_S = 67,65
592	R.M.D.= 627,52 C.E. = 135,43 C.E.nR. = 135,43 C.L.W.= 7,24 M_F = 3,04 T_S = 67,19	R.M.D.= 448,39 C.E.= 14,55 C.E.R.= 38,74 C.E.nR. = 5,55 C.L.W.= 1,75 M_F = 1,82 T_S = 67,60	R.M.D.= 229,01 C.E.= 12,09 C.E.R.= 30,28 C.E.nR. = 2,03 C.L.W.= 1,08 M_{SR}= 0,61 MP_{SR}= 9 M_F = 1,36 T_S = 65,64
800	R.M.D.= 755,79 C.E. = 193,79 C.E.nR. = 193,79 C.L.W.= 9,74 M_F = 3,61 T_S = 67,67	R.M.D.= 451,15 C.E.= 18,05 C.E.R.= 51,85 C.E.nR. = 5,48 C.L.W.= 1,73 M_F = 1,80 T_S = 67,58	R.M.D.= 246,42 C.E.= 15,65 C.E.R.= 40,29 C.E.nR. = 2,04 C.L.W.= 1,08 M_{SR}= 0,61 MP_{SR}= 10 M_F = 1,38 T_S = 66,65

R.M.D.= Retardo Medio de Distribución (ms)
C.E.= Carga por Encaminador (paquetes)
C.E.R.= Carga por Encaminador RMNP (paquetes)
C.E.nR.= Carga por Encaminador no RMNP (paquetes)
C.L.W.= Carga por Línea de la WAN (bytes)
M_{SR}= Memoria por SR de la WAN (paquetes)
MP_{SR}= Memoria de pico en los SRs (paquetes)
M_F= N° medio de paquetes almacenados en la fuente (paquetes)
T_S= Tiempo simulado (minutos)

Tabla 6.6: Pruebas realizadas variando el número de miembros (R_{MP}= 0,4)

Configuración / Nº miembros	Configuración1	Configuración2	Configuración3
16	R.M.D.= 545,30 C.E. = 5,04 C.E.nR. = 5,04 C.L.W.= 1,81 M_F = 2,95 T_S = 34,66	R.M.D.= 509,57 C.E.= 4,97 C.E.R.= 2,67 C.E.nR. = 5,83 C.L.W.= 1,85 M_F = 3,29 T_S = 34,65	R.M.D.= 226,5 C.E.= 2,29 C.E.R.= 2,74 C.E.nR. = 2,04 C.L.W.= 1,09 M_{SR}= 1,05 MP_{SR}= 10 M_F = 2,34 T_S = 33,57
192	R.M.D.= 607,25 C.E. = 44,11 C.E.nR. = 44,11 C.L.W.= 3,48 M_F = 3,93 T_S = 35,35	R.M.D.= 515,98 C.E.= 7,93 C.E.R.= 13,68 C.E.nR. = 5,79 C.L.W.= 1,84 M_F = 3,26 T_S = 35,63	R.M.D.= 227,84 C.E.= 5,28 C.E.R.= 11,13 C.E.nR. = 2,04 C.L.W.= 1,09 M_{SR}= 1,05 MP_{SR}= 10 M_F = 2,35 T_S = 33,37
400	R.M.D.= 682,77 C.E. = 94,66 C.E.nR. = 94,66 C.L.W.= 5,64 M_F = 4,66 T_S = 35,81	R.M.D.= 537,22 C.E.= 11,51 C.E.R.= 26,75 C.E.nR. = 5,84 C.L.W.= 1,86 M_F = 3,33 T_S = 35,72	R.M.D.= 225,59 C.E.= 8,82 C.E.R.= 21,09 C.E.nR. = 2,04 C.L.W.= 1,09 M_{SR}= 1,06 MP_{SR}= 10 M_F = 2,36 T_S = 33,24
592	R.M.D.= 955,71 C.E. = 157,02 C.E.nR. = 157,02 C.L.W.= 8,42 M_F = 6,10 T_S = 38,32	R.M.D.= 540,85 C.E.= 14,81 C.E.R.= 38,89 C.E.nR. = 5,86 C.L.W.= 1,86 M_F = 3,39 T_S = 35,33	R.M.D.= 224,31 C.E.= 12,10 C.E.R.= 30,30 C.E.nR. = 2,04 C.L.W.= 1,09 M_{SR}= 1,06 MP_{SR}= 10 M_F = 2,35 T_S = 33,48
800	R.M.D.= 4388,22 C.E. = 438,67 C.E.nR. = 438,67 C.L.W.= 21,93 M_F = 9,18 T_S = 85,01	R.M.D.= 522,80 C.E.= 18,35 C.E.R.= 52,00 C.E.nR. = 5,83 C.L.W.= 1,85 M_F = 3,34 T_S = 35,61	R.M.D.= 233,25 C.E.= 15,67 C.E.R.= 40,32 C.E.nR. = 2,04 C.L.W.= 1,09 M_{SR}= 1,06 MP_{SR}= 10 M_F = 2,36 T_S = 34,01

R.M.D.= Retardo Medio de Distribución (ms)
C.E.= Carga por Encaminador (paquetes)
C.E.R.= Carga por Encaminador RMNP (paquetes)
C.E.nR.= Carga por Encaminador no RMNP (paquetes)
C.L.W.= Carga por Línea de la WAN (bytes)
M_{SR}= Memoria por SR de la WAN (paquetes)
MP_{SR}= Memoria de pico en los SRs (paquetes)
M_F= Nº medio de paquetes almacenados en la fuente (paquetes)
T_S= Tiempo simulado (minutos)

Tabla 6.7: Pruebas realizadas variando el número de miembros (R_{MP}= 0,2)

A continuación se muestran gráficamente algunos de los resultados obtenidos:

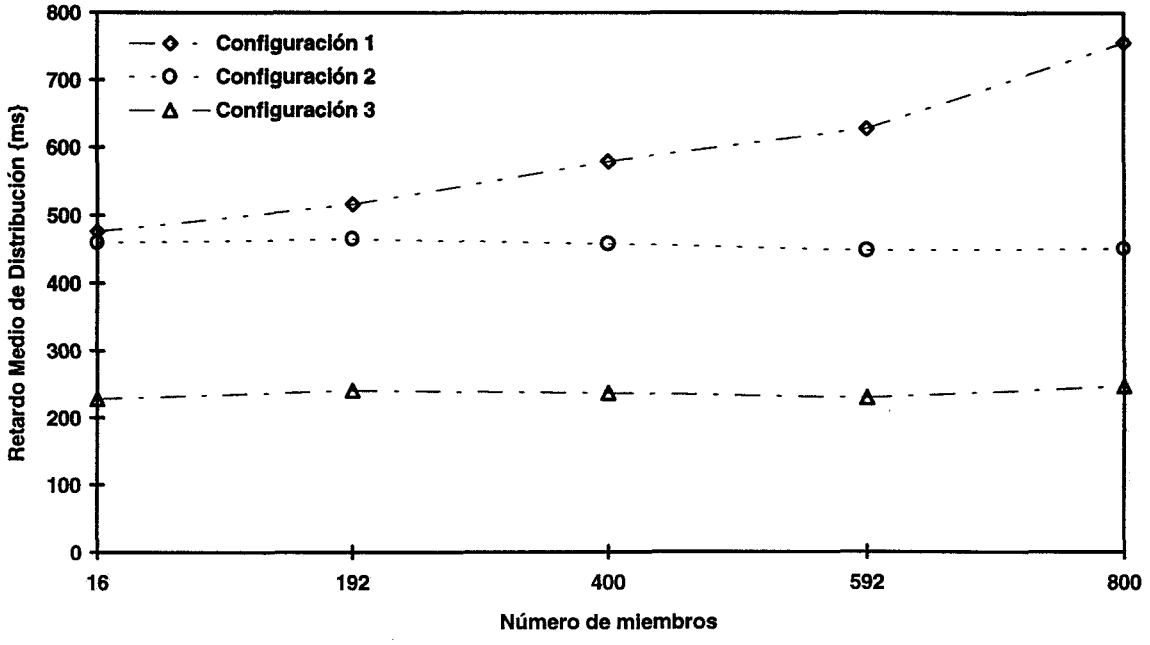


Figura 6.16: R.M.D. versus número de miembros (R_{MP}= 0,4 sg.)

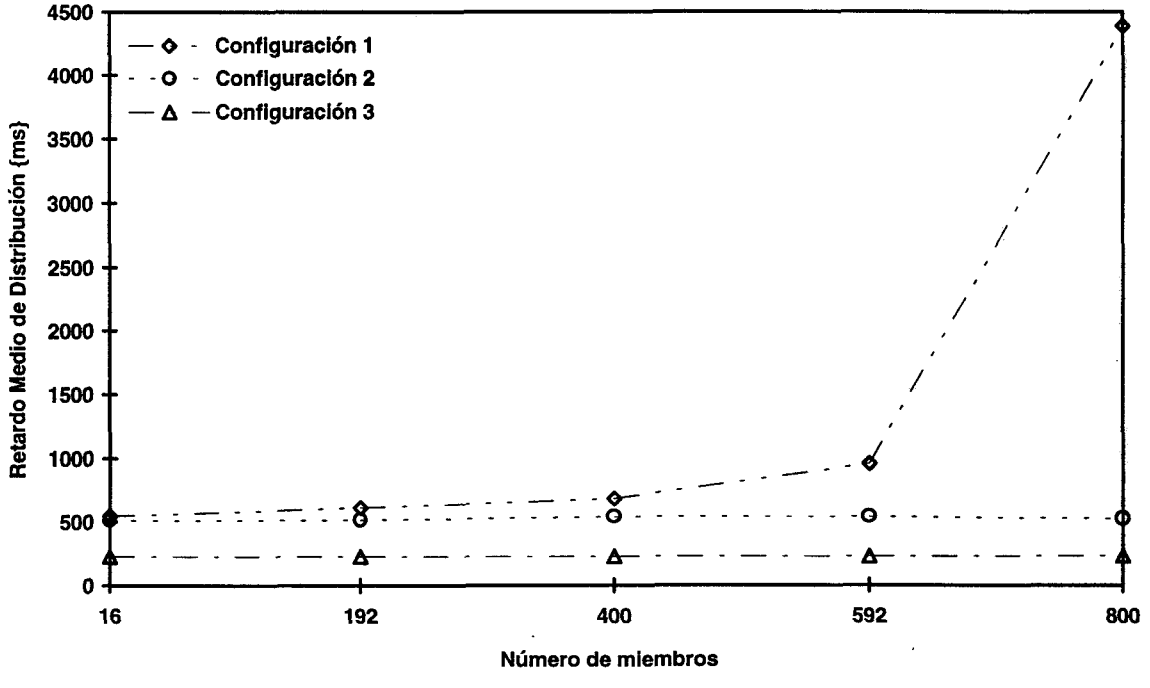


Figura 6.17: R.M.D. versus número de miembros (R_{MP}= 0,2 sg.)

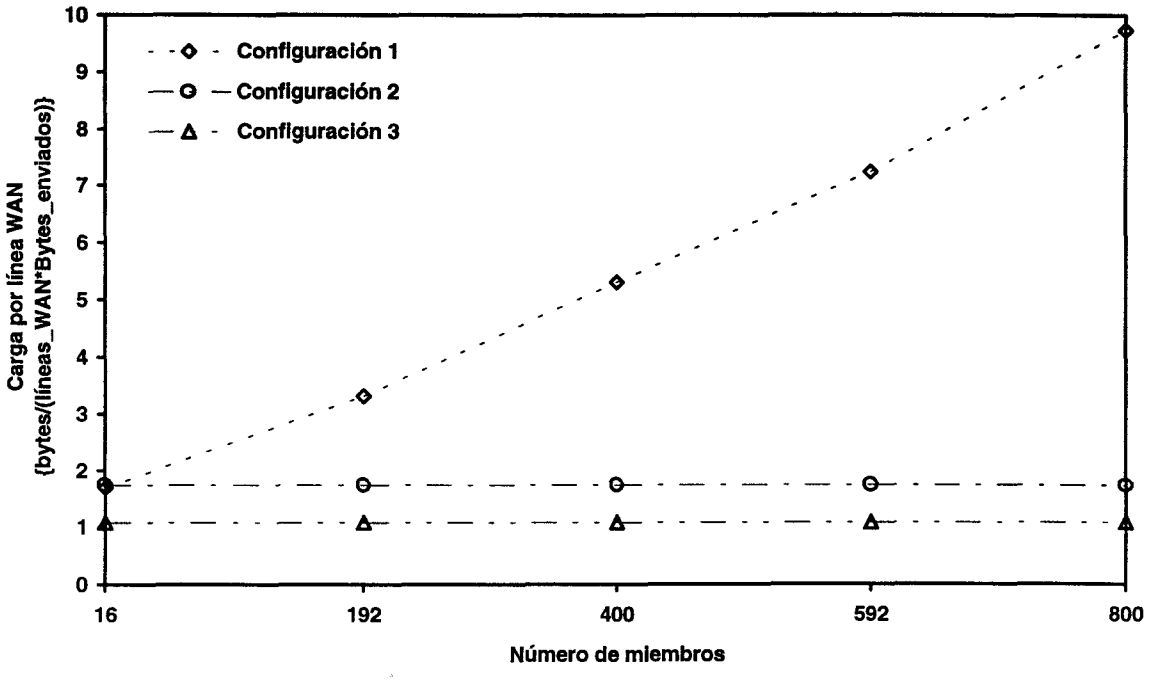


Figura 6.18: C.L.W. versus número de miembros (R_{MP}= 0,4)

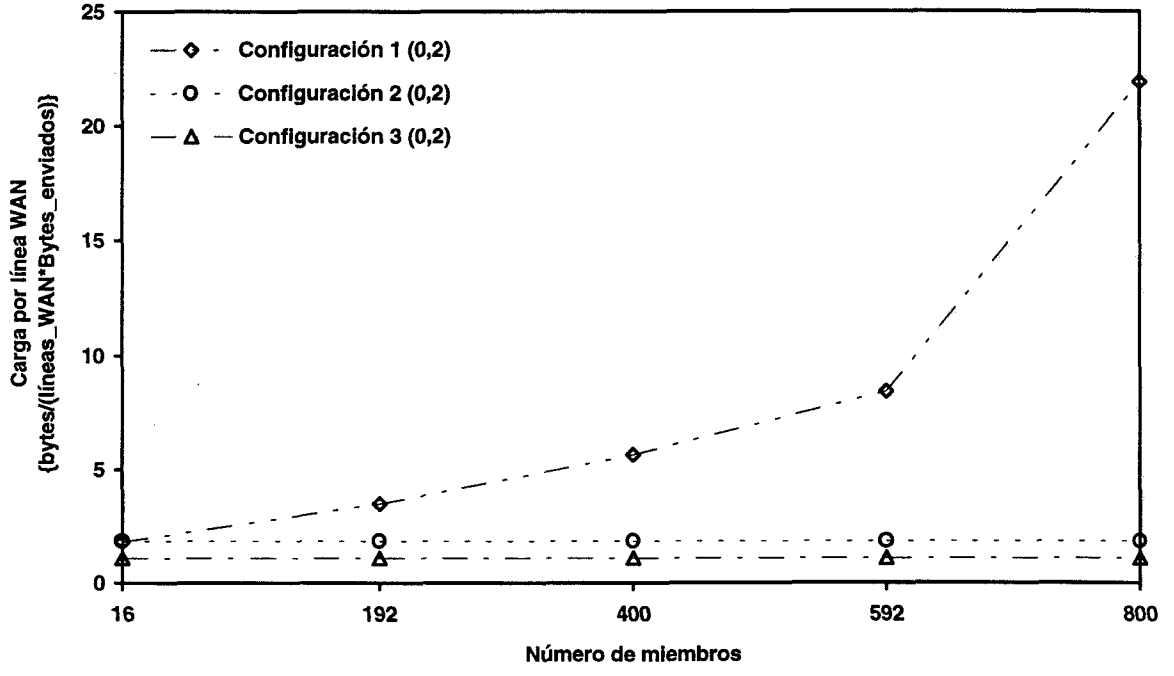


Figura 6.19: C.L.W. versus número de miembros (R_{MP}= 0,2)

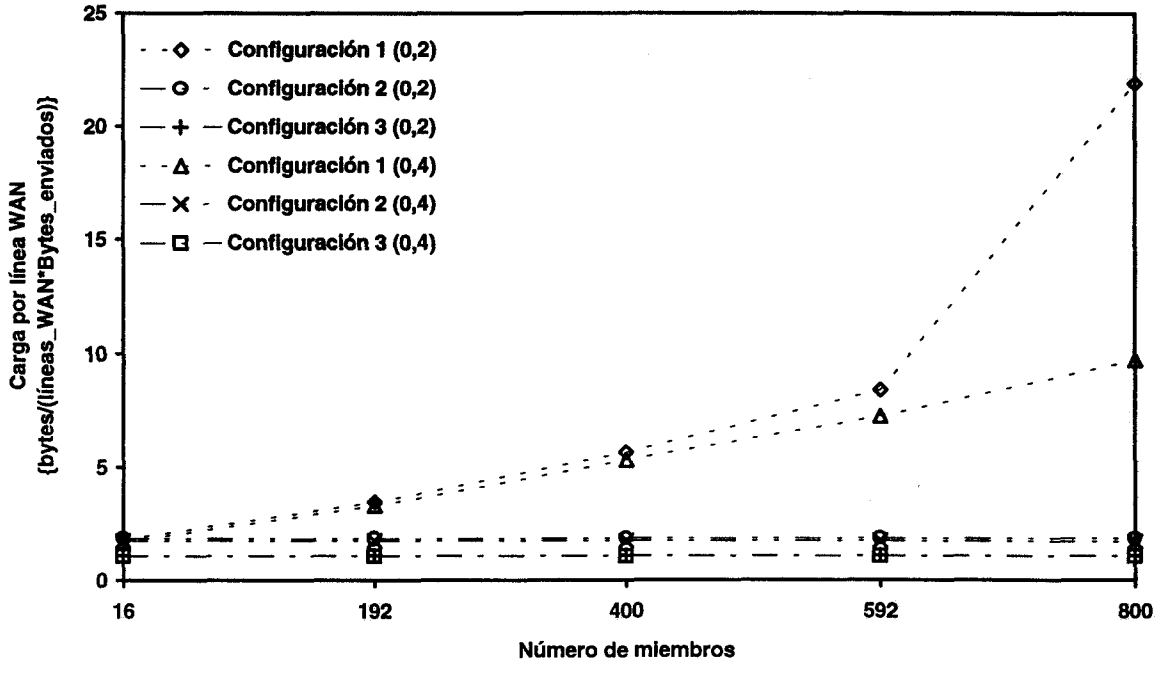


Figura 6.20: C.L.W. versus número de miembros

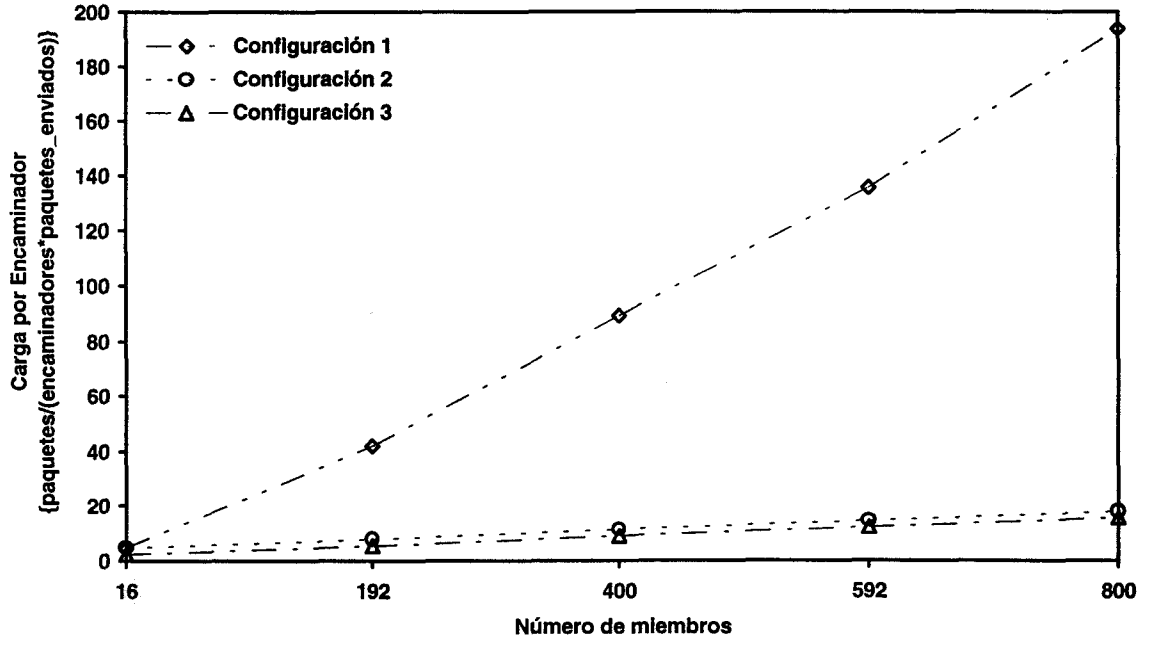


Figura 6.21: C.E. versus número de miembros (R_{MP}= 0,4)

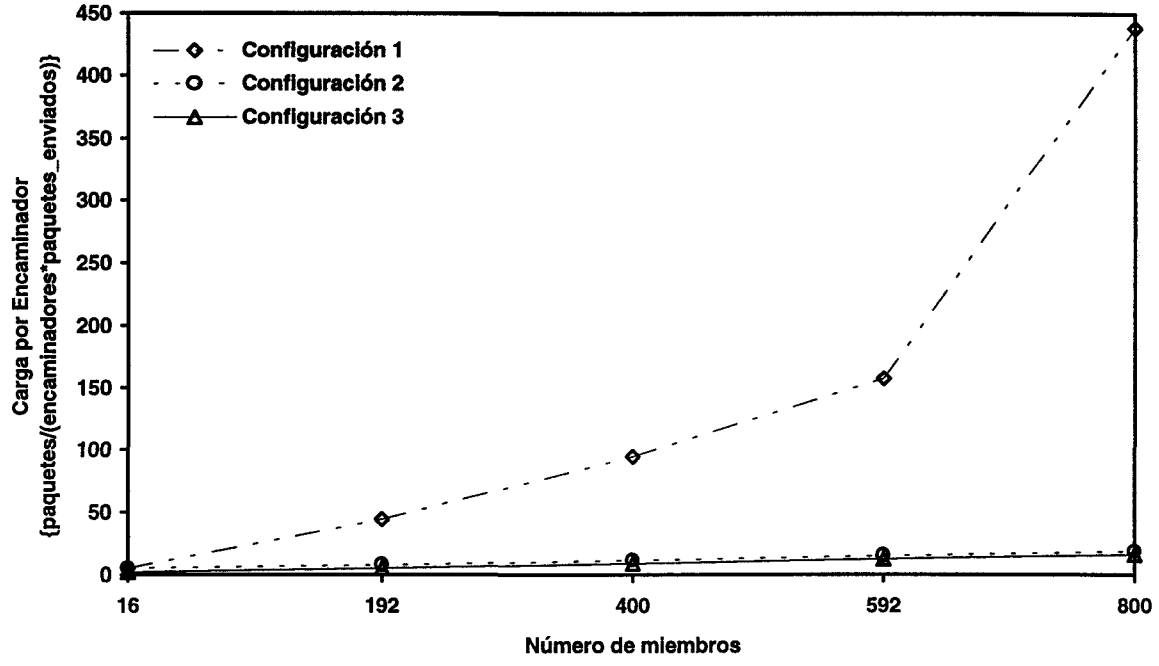


Figura 6.22: C.E. versus número de miembros (R_{MP}= 0,2)

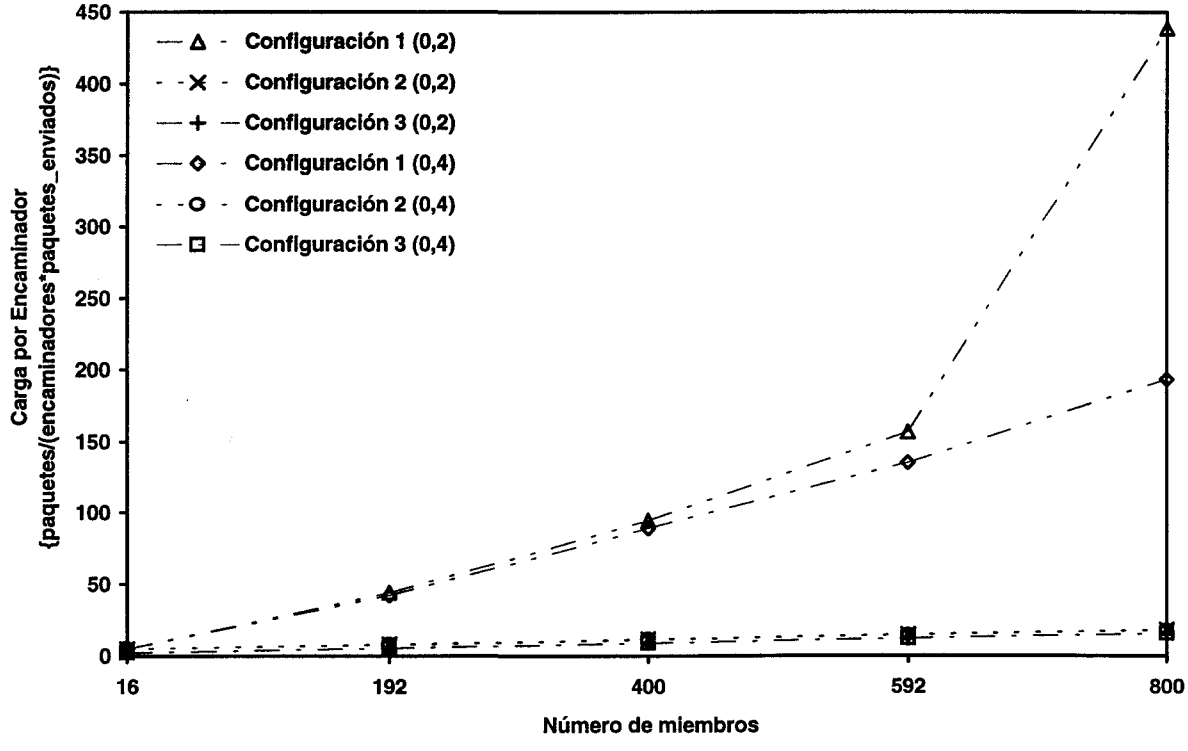


Figura 6.23: C.E. versus número de miembros

6.7 Interpretación de los resultados

Se considera interesante reseñar que al realizar las primeras pruebas de simulación, se eligió un valor de 0.1 sg. como R_{MP}. Dado este valor, se estaban introduciendo en la red una media de 10 nuevos paquetes de datos por segundo, y considerando el tamaño de los paquetes de datos (1088 bytes), el caudal medio era de aproximadamente 10 Kbytes/sg. Aunque este caudal se puede considerar moderado y la capacidad de las líneas simuladas aceptable (2 Mbps las líneas de la WAN y 100 Mbps las líneas de los campus), sin embargo se producía un problema de congestión en los enlaces y encaminadores cercanos a la fuente. Dicha congestión era debida al problema de la *implosión de ACKs*. Debe tenerse en cuenta que en una configuración sin encaminadores RMNP y con 400 miembros (configuración 1), si la fuente introduce en la red 10 nuevos paquetes de datos cada segundo, estos dan lugar a 4000 ACKs/sg. Dado el tamaño de un ACK (44 bytes), el enlace que une el campus de la fuente con la WAN debe cursar 1,4 Mbps, adicionalmente a este tráfico de ACKs se le debe incrementar tanto el tráfico de NAKs, como el de ACKs asociados a paquetes previamente confirmados y que han sido retransmitidos. Dado lo anterior, es explicable que con dichos parámetros se produzca congestión en la red, aumentando de forma progresiva el número de paquetes circulando por la red, y al mismo tiempo que el R.M.D. principalmente, pero también el C.L.W. y C.E., vayan creciendo de forma desorbitada. Esta experiencia corrobora la idea de que los protocolos multipunto fiables que siguen una aproximación orientada al emisor, y no incorporan ningún mecanismo de agregación, no escalan con relación al tamaño del grupo.

Por otro lado, con el mismo caudal de entrada de 10 Kbytes/sg (R_{MP}= 0,1 sg.) también se detectaron problemas de congestión en las configuraciones del tipo 1, al aumentar el tamaño de la WAN. Esto es debido a que al aumentar el tamaño de la WAN se producen más pérdidas de paquetes (hay más saltos WAN) y consiguientemente se realizan más retransmisiones. Dichas pérdidas y las correspondientes retransmisiones, hacen que aumente el flujo de NAKs y ACKs hacia la fuente. Recuérdese que cuando un miembro recibe un paquete previamente confirmado vuelve a enviar un ACK por si acaso el anterior se perdió.

Con el fin de ilustrar lo anteriormente comentado la Figura 6.24 muestra, utilizando una configuración del tipo 1 con 400 miembros y R_{MP}= 0,5 sg., el estado de los buffers de salida del encaminador que une el campus de la fuente a la WAN (encaminador fuente), presentándose tanto el buffer de la línea hacia la fuente como el buffer de la línea hacia la WAN (los miembros). Como puede observarse, y aunque el caudal introducido no es muy elevado (aproximadamente 2 Kbytes/sg.), el tráfico de paquetes de control con destino a la fuente es muy significativo en comparación con los datos. De forma semejante en la Figura

6.25 se muestran dos gráficas, una con el número total de paquetes circulando por la red en un instante dado y otra con el número de paquetes de datos, y puede verse que los paquetes de datos constituyen una parte muy pequeña con relación al número total de paquetes. Consiguientemente, dado que en la gran mayoría de las pruebas de simulación realizadas el número de miembros es relativamente elevado, el tráfico de ACKs y NAKs que fluyen hacia la fuente, constituye un factor decisivo a la hora de interpretar los resultados.

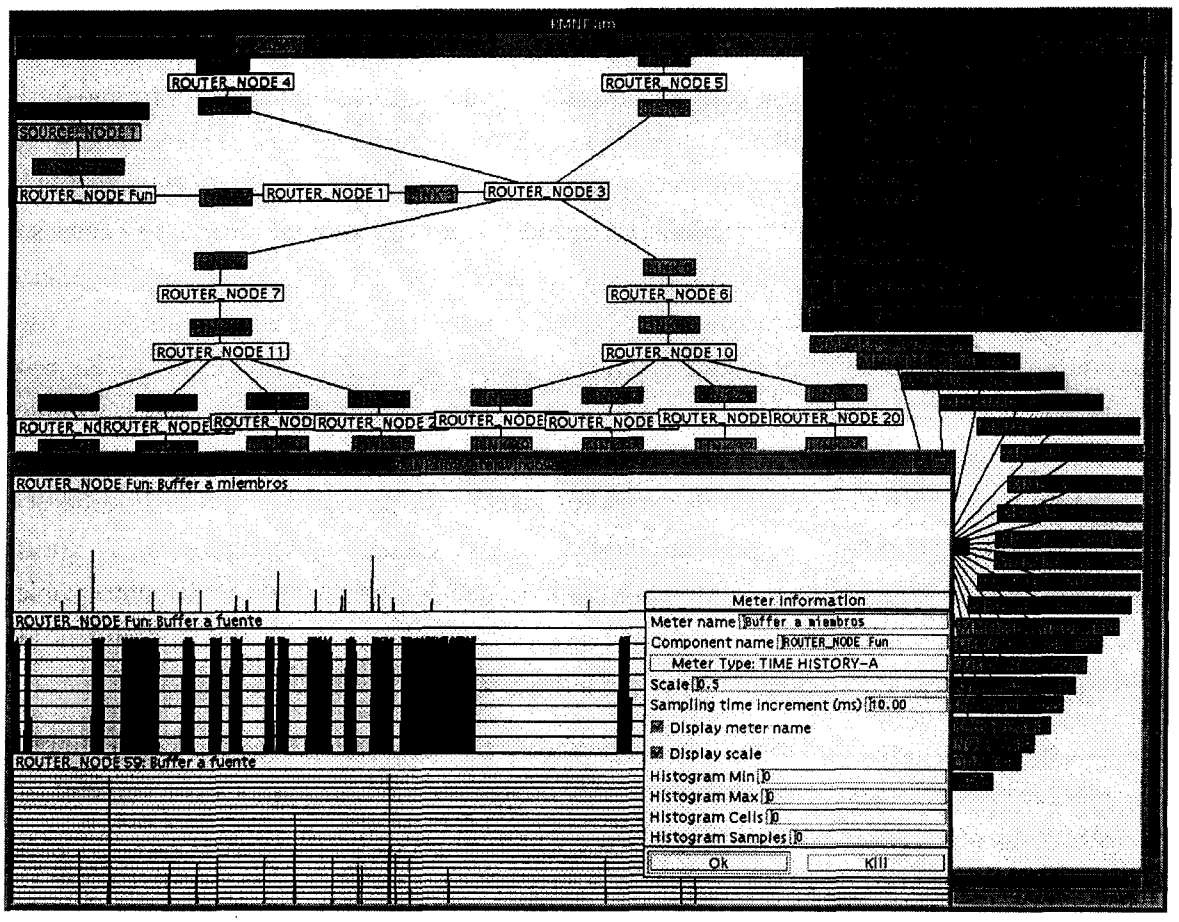


Figura 6.24: Evolución de los buffers del encaminador fuente

Visto todo lo anterior se decidió aumentar el R_{MP} para con esto disminuir el caudal de datos introducido en la red y evitar que se produjeran problemas de congestión al realizar las simulaciones de las configuraciones del tipo 1. En concreto en la batería de pruebas donde se varía el número de miembros se utilizó un R_{MP} de 0,4 sg., de modo que el caudal se redujo a 2,5 Kbytes/sg (ver Tabla 6.6). Del mismo modo al variar el tamaño de la WAN se eligió un R_{MP} de 0,5 sg., lo que da lugar a un caudal de 2 Kbytes/sg (ver Tabla 6.4).

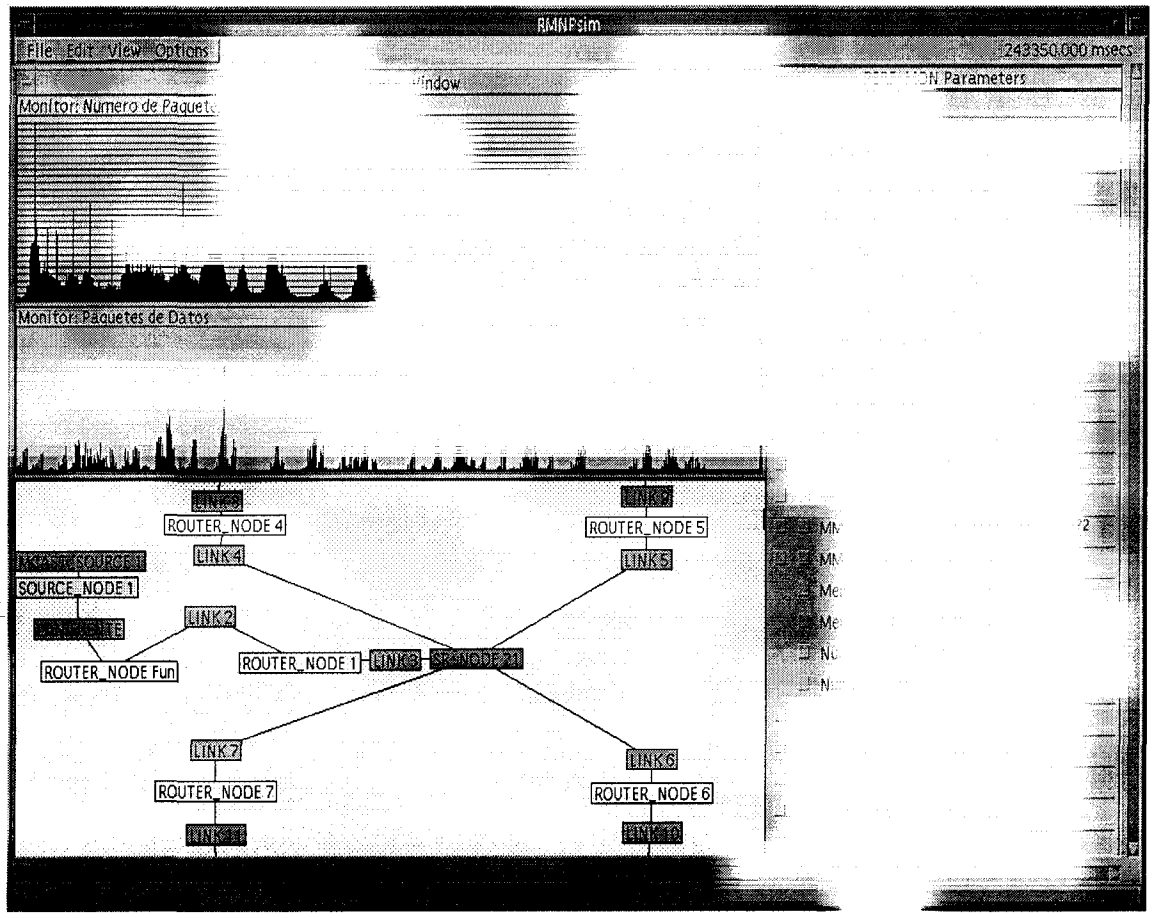


Figura 6.25: Evolución del número de paquetes en red

Adicionalmente y con la finalidad de mostrar que en las configuraciones de los tipos 2 y 3 con caudales superiores no se producen problemas de congestión se han repetido ambas baterías de pruebas con un R_{MP} inferior. En la Tabla 6.7 se presentan los resultados de variar el número de miembros utilizando un R_{MP} igual a 0,2 sg. y en dicha tabla puede observarse que en la configuración tipo 1 correspondiente a 800 miembros, el tiempo empleado para enviar los mismos 10.000 paquetes de datos es de 85,01 minutos, mientras que en el resto de configuraciones el tiempo medio empleado es de aproximadamente 35 minutos. Este dato constituye un indicio claro de que en dicha prueba la red estaba congestionada. Como puede observarse, al producirse congestión en las configuraciones de tipo 1 se disparan los valores de R.M.D. (ver gráfica 6.17), C.L.W. (ver gráficas 6.19 y 6.20) y C.E. (ver gráficas 6.22 y 6.23). Esto es debido a que como consecuencia de la congestión aumenta significativamente el tiempo que tardan los ACKs y NAKs en llegar a la fuente, lo cual tiene tres consecuencias: (1) la ventana en la fuente se desplaza más lentamente y consiguientemente el número medio de paquetes almacenados en la ventana aumenta sensiblemente (ver valores de M_F) y debe recordarse que cada vez que se hace una retransmisión se reenvía la ventana completa; (2) los procesos de retransmisión tardan más tiempo

en ponerse en marcha; y (3) es relativamente fácil que se produzcan vencimientos prematuros del temporizador de retransmisión. Como consecuencia de todo esto, cuando la red se congestiona se disparan los valores de R.M.D., C.L.W. y C.E. Adicionalmente, en dichas gráficas también puede observarse que con dicho caudal para las configuraciones 2 y 3 no se produce ningún problema de congestión.

Análogamente, en la Tabla 6.5 se presentan los resultados de variar el tamaño de la WAN utilizando un R_{MP} igual a 0,4 sg. y siguiendo los mismos razonamientos antes expuestos puede concluirse que con dicho caudal en la WAN 5 al emplear una configuración del tipo 1 se producen problemas de congestión. Asimismo puede verse, igual que antes y por las mismas razones, que se disparan los valores de R.M.D. (ver gráfica 6.9), C.L.W. (ver gráficas 6.11 y 6.12) y C.E. (ver gráficas 6.14 y 6.15), mientras que en las configuraciones de los tipos 2 y 3 para el mismo caudal no se detectan problemas de congestión. Debe reseñarse que si el caudal de datos hubiera sido superior los problemas de congestión se habrían empezado a detectar antes, o lo que es lo mismo, en configuraciones con menos miembros o en redes más pequeñas.

A continuación se pasa a analizar de forma separada los valores obtenidos para cada uno de los parámetros y para esto se tendrán en cuenta los resultados obtenidos en las pruebas de simulación con caudal más bajo y consecuentemente sin problemas de congestión (Tablas 6.4 y 6.6).

6.7.1 Análisis del retardo de distribución

Los resultados relativos al **R.M.D.** tanto al variar el tamaño de la interred como el número de miembros, se muestran en las gráficas 6.8 y 6.16 respectivamente. Con respecto al R.M.D. se consideran dignos de mención los siguientes aspectos:

- En todas las pruebas de simulación realizadas los menores retardos de distribución se han obtenido en las configuraciones de tipo 3, llegando al extremo de que en la WAN 5, el R.M.D. es aproximadamente 6 veces inferior en la configuración 3 que en la configuración 1. Esto se debe principalmente a que en las configuraciones de tipo 3 la mayoría de las retransmisiones se realizan desde un SR en lugar de realizarse desde la fuente.

- Los valores de las configuraciones de tipo 2 son muy cercanos a los obtenidos para las configuraciones de tipo 1. Esto tiene por justificación que dado que la pérdida de paquetes es mucho más probable en la WAN que en los campus y que en las

configuraciones de tipo 2 no existen SRs en la WAN, la gran mayoría de las retransmisiones se hacen, al igual que en las configuraciones tipo 1, desde la fuente. Sin embargo, en las configuraciones de tipo 2 los SRs de los campus realizan agregación de ACKs y de aquí que los NAKs tarden menos tiempo en llegar a la fuente obteniéndose mejores resultados en éstas que en las configuraciones de tipo 1.

- Al variar el *tamaño de la WAN*, el R.M.D. crece en los tres tipos de configuraciones, esto es simplemente porque la fuente y los miembros están más alejados. Sin embargo, el crecimiento para las configuraciones tipo 3 es mucho más suave que para los otros dos tipos, de modo que para interredes muy extensas la configuración 3 es la única que ofrece valores aceptables de R.M.D.

- Al variar el *número de miembros*, el R.M.D. de las configuraciones de los tipos 2 y 3 es prácticamente constante, de modo que las configuraciones de ambos tipos escalan adecuadamente con respecto al número de miembros. Por el contrario en los resultados para la configuración 1 puede observarse que el retardo se incrementa al crecer el número de miembros, esto es debido a que en la misma proporción que el número de miembros aumenta el número de ACKs y NAKs que fluyen hacia la fuente y consecuentemente éstos últimos tardan más tiempo en llegar a su destino. Esto da lugar a que la fuente ponga en marcha más tarde los procesos de retransmisión, incrementándose el retardo.

  Puede resultar extraño y merece la pena ser comentado que en las configuraciones de tipo 1, el R.M.D. crece pero levemente al aumentar el número de miembros. Lo normal es que al aumentar el número de miembros, si estos están dispersos, se incremente el número de saltos WAN atravesados y consecuentemente aumente la probabilidad de que los paquetes se pierdan y deban ser retransmitidos. Este aumento en el número de retransmisiones provoca un incremento en el R.M.D., el C.E. y el C.L.W. y obviamente, este efecto se acentúa en las configuraciones equivalentes a las del tipo 1 donde las retransmisiones se hacen desde la fuente. Sin embargo, en las pruebas de simulación aquí realizadas los miembros siempre se agrupan en los mismos 16 campus (el número de saltos WAN es constante haya 16 u 800 miembros) y de aquí que no aumente significativamente el número de retransmisiones. Simplemente existe un leve aumento en el número de retransmisiones originado porque al incrementarse el número de miembros se aumenta la probabilidad de que algún miembro descarte un paquete, sin embargo esto no introduce un cambio apreciable en el retardo, ya que la probabilidad de que un miembro descarte un paquete por falta de buffers es muy baja.

- Después de realizar distintas pruebas de simulación, se ha observado que el R.M.D. es un valor que no llega a estabilizarse totalmente aunque el tiempo simulado sea muy grande y especialmente en las configuraciones con muchos miembros o con interredes muy extensas. Esto se debe a que por su propia definición (6.3), como el tiempo medio transcurrido desde que un paquete es liberado por la fuente hasta que es recibido correctamente por *todos* los miembros del grupo, en el R.M.D. tienen mucho peso los valores atípicos. Esto es, un miembro que tarda un tiempo muy elevado en recibir un paquete mientras que la gran mayoría de los miembros lo recibieron mucho antes o un paquete que tiene un retardo de distribución altísimo con respecto al resto de paquetes (por ejemplo, en la configuración 1 WAN 1 el R.M.D. es 1.249,78 ms mientras que el máximo retardo de distribución[2] que ha sufrido un paquete ha sido 12.773,91 ms).

6.7.2 Análisis de la carga por línea WAN

Los resultados relativos a la **C.L.W.** tanto al variar el tamaño de la interred como el número de miembros se muestran en las gráficas 6.10 y 6.18 respectivamente. Con respecto a la C.L.W. se consideran dignos de comentar los siguientes aspectos:

- Los mejores resultados respecto a la C.L.W. son los obtenidos para las configuraciones de tipo 3, llegando al extremo de que en la WAN 5, la C.L.W. es aproximadamente 9 veces inferior en la configuración 3 que en la configuración 1.

- En las configuraciones de tipo 3, tanto al variar el número de miembros como el tamaño de la interred, se observa que los valores de C.L.W. se mantienen prácticamente invariables en el rango [1 ,2] bytes. Esto es gracias a las funciones de agregación de ACKs y filtrado de NAKs, y al sistema de retransmisiones locales.

- En las configuraciones de tipo 1, tanto al variar el número de miembros como el tamaño de la interred, se observa un incremento muy significativo en los valores de C.L.W. Adicionalmente debe tenerse en consideración que por las mismas razones expuestas al analizar el R.M.D., la C.L.W. aumentaría más deprisa con respecto al número de miembros, si dichos miembros estuvieran más dispersos en lugar de concentrarse en los 16 campus.

[2] El máximo retardo de distribución es otro de los parámetros medidos por el RMNPsim. Este parámetro concreto no se ha mostrado en las tablas de los resultados obtenidos por no considerarse muy relevante.

- En las configuraciones de tipo 2 puede observarse que al aumentar el número de miembros la C.L.W. se mantiene prácticamente constante a 1,75 bytes, esto es debido a que dadas las configuraciones elegidas, la principal consecuencia al aumentar el número de miembros es que se produce un mayor número de ACKs mientras que el número de retransmisiones se mantiene prácticamente constante, y las configuraciones de tipo 2 son insensibles a dicho aumento de ACKs gracias a la función de agregación en los campus. Por el contrario, un aumento del tamaño de la interred se caracteriza principalmente por un incremento en el número de retransmisiones. Dicho incremento en las retransmisiones causa que en las configuraciones de tipo 2 aumente la C.L.W. al variar la interred. Nótese que al medir la C.L.W. los paquetes de datos, aunque no sean muy numerosos con relación al número de paquetes de control, son decisivos por su tamaño, un paquete de datos equivale a 25 paquetes de control. A pesar de todo lo anterior los valores obtenidos para las configuraciones de tipo 2 siempre están muy por debajo de los obtenidos para las configuraciones de tipo 1, gracias a que aunque en ambos casos la mayoría de retransmisiones se realizan desde la fuente, en las configuraciones de tipo 2 se realiza la función de agregación en los campus.

6.7.3 Análisis de la carga por encaminador

Los resultados relativos a la **C.E.** tanto al variar el tamaño de la interred como el número de miembros se muestran en las gráficas 6.13 y 6.21 respectivamente. Con respecto a la C.E. se consideran dignos de mencionar los siguientes aspectos:

- Los mejores resultados respecto a la C.E. son los obtenidos para las configuraciones de tipo 3, llegando al extremo de que en la WAN 5, la C.E. es aproximadamente 20 veces inferior en la configuración 3 que en la configuración 1.

- En las configuraciones de tipo 1, tanto al variar el número de miembros como el tamaño de la interred, se observa un incremento muy importante en los valores de C.E., siendo especialmente significativo al crecer el tamaño del grupo. Adicionalmente debe tenerse en consideración que por las mismas razones expuestas al analizar el R.M.D., la C.E. aumentaría más deprisa con respecto al número de miembros, si dichos miembros estuvieran más dispersos en lugar de concentrarse en los 16 campus.

- En las configuraciones de los tipos 2 y 3 se observa una fluctuación muy leve de la C.E. que en todas las simulaciones se ha mantenido por debajo de 20 paquetes.

Ambos tipos de configuraciones tienen un comportamiento bastante similar pues en ambos casos existen SRs en los campus que realizan la función de agregación de ACKs, lógicamente la C.E. es algo superior en las configuraciones de tipo 2 pues en este caso no existen SRs en la WAN. Si en lugar de analizar la C.E. se estudia el comportamiento de los valores de la C.E.R. y la C.E.nR. se puede observar que la carga en los encaminadores no RMNP se mantiene prácticamente constante (al variar el número de miembros) o aumenta ligeramente (al variar el tamaño de la interred) pero esto es a costa de un mayor proceso en los encaminadores RMNP, principalmente al variar el número de miembros pues en estas pruebas es donde se produce un gran incremento en el número de paquetes de control.

6.7.4 Análisis de los requisitos de memoria

En bastantes de las pruebas de simulación realizadas, la MP_{SR} coincide con el tamaño de la ventana en la fuente, sin embargo puede observarse que el número medio de los paquetes almacenados en los SRs de la WAN[3] (M_{SR}) es muy bajo estando por debajo de la memoria precisada en la fuente; un dato significativo es que en todas las simulaciones realizadas el valor de M_{SR} es inferior a la mitad de M_F. Para interpretar adecuadamente estos datos en primer lugar debe tenerse en cuenta que el valor MP_{SR} mide la máxima necesidad de memoria de alguno de los SRs de la WAN durante toda la comunicación, pero que si en lugar de esto se hubiese calculado la media de los valores de pico en los SRs de la WAN los valores obtenidos hubieran sido inferiores. Por otro lado, dado que un mismo SR formará parte de varios árboles RMNP y que los encaminadores tienen asignación dinámica de memoria no constituye un gran problema que durante un corto periodo de tiempo las necesidades de memoria para un determinado árbol RMNP sean altas, lo realmente significativo es que el número medio de paquetes almacenados en los SRs de la WAN es bajo.

6.7.5 Conclusiones

De las pruebas de simulación realizadas se han extraído las siguientes conclusiones:

- El RMNP exhibe un comportamiento muy satisfactorio tanto al aumentar el tamaño del grupo como al aumentar el tamaño de la interred y de aquí que se pueda afirmar que el RMNP escala adecuadamente en ambas dimensiones.

[3] Esta media se ha calculado cada 25 ms.

- Los protocolos de multipunto fiable que operan a nivel de transporte y siguen una aproximación orientada al emisor tradicional no escalan adecuadamente ni con relación al tamaño del grupo ni respecto al tamaño de la interred.

- Si el número de miembros es elevado y está agrupado en un cierto número de campus pero la interred no es muy extensa, normalmente será suficiente disponer de SRs únicamente en los campus para obtener unas prestaciones aceptables, al mismo tiempo que la solución será más barata que disponer de SRs en los campus y en la WAN.

- Si la interred es muy extensa o la probabilidad de pérdida de paquetes es elevada, resulta fundamental introducir SRs en la WAN para que el retardo medio de distribución sea aceptable y especialmente para las aplicaciones sensibles al retardo.

Capítulo 7.

ANÁLISIS COMPARATIVO

En este capítulo se va a realizar un análisis comparativo de los mecanismos fundamentales de los protocolos de multipunto fiable, comparándose la solución adoptada por el RMNP frente a las de otros protocolos de multipunto fiable. Siempre teniendo en mente que los objetivos de diseño del RMNP son que: (1) minimice el retardo de distribución, (2) sea escalable con relación al número de miembros y al tamaño de la interred, (3) haga un buen uso de los recursos de red, y (4) logre un grado de fiabilidad aceptable para un amplio rango de aplicaciones.

7.1 Esquema de retransmisiones

Los protocolos de multipunto fiable siguen distintas aproximaciones al plantear *desde qué punto* se realizan las retransmisiones. Las distintas soluciones pueden clasificarse en *esquemas de retransmisión centralizados y distribuidos.* En el primer caso, las retransmisiones se realizan desde un único punto mientras que en el segundo, las retransmisiones se realizan desde *n* puntos distintos atendiendo a distintas consideraciones, siendo la más usual la proximidad geográfica.

El RMNP utiliza un esquema de retransmisión distribuido, ya que tanto la fuente como los SRs almacenan todos los paquetes pendientes de confirmación y son capaces de llevar a cabo procesos de retransmisión. En concreto, en RMNP la recuperación ante errores se hace desde el sistema retransmisor (fuente o SR) más cercano en el árbol RMNP al punto donde se originó el problema.

La gran mayoría de los protocolos de multipunto fiable utilizan esquemas centralizados, en los cuales las retransmisiones se realizan desde un único punto que puede ser, o bien la fuente como en el caso del MTP [AFM92], el XTP [Xpr95] o el FTM [Raj91], o bien un sistema con responsabilidades especiales, como por ejemplo el receptor primario en el caso del RMP [Whe95] o el secuenciador [KTHB89]. El principal problema de este tipo de soluciones es que escalan mal con relación al tamaño del grupo, ya que el sistema retransmisor puede convertirse en un "cuello de botella", al tener que procesar todas las solicitudes y realizar todas las retransmisiones. Por el contrario cuando se utilizan esquemas distribuidos (1) se disminuye el retardo medio de distribución al realizar las retransmisiones desde un sistema más cercano al punto donde se originó el problema, (2) se hace un buen uso de los recursos de red, y (3) se permite la recuperación ante diferentes errores de forma concurrente e independiente, en distintas partes de la interred. Estas ventajas ya han sido analizadas de manera cuantitativa con las pruebas de simulación presentadas en el capítulo 6, ya que se han comparado los resultados obtenidos en la configuración 1 donde se ha utilizado un esquema de retransmisiones centralizado al realizarse todas las retransmisiones desde la fuente por no existir SRs, y la configuración 3 donde se ha utilizado un esquema de retransmisiones distribuido al disponer de SRs en distintos puntos de la interred.

El principal inconveniente de los esquemas distribuidos es que los n sistemas retransmisores deben mantener copia de todos los paquetes ya enviados al grupo y que podrían ser solicitados (por ejemplo, los paquetes pendientes de confirmación); hecho que tiene como consecuencia que se precise más memoria en buffers. Sin embargo como ya se ha visto en el capítulo de simulación, en RMNP las necesidades de memoria de los sistemas retransmisores pueden calificarse de moderadas.

Ciertos trabajos simultáneos al RMNP como el SRM [FJM+95], el TMTP [YGS95] o el RMTP [LS96] también utilizan esquemas de retransmisión distribuidos, pero sin embargo éstos tratan de resolver el problema a nivel de transporte (la retransmisión se hace desde otro miembro, cercano al que solicitó la retransmisión). Consecuentemente, al no utilizar el conocimiento de la red sobre la topología del grupo, se ven obligados a introducir complejos y costosos mecanismos para discernir en cada situación, cual es el miembro más cercano que está en condiciones de realizar la retransmisión (tiene disponibles los paquetes solicitados).

7.2 Ámbito de las retransmisiones

Los protocolos de multipunto fiable siguen distintas aproximaciones al plantear *el alcance de los paquetes retransmitidos.* Las distintas soluciones pueden clasificarse en *esquemas de retransmisión de ámbito global y restringido (o local).*

Al utilizarse un esquema de ámbito global, cuando se solicita la retransmisión de uno o más paquetes, el sistema retransmisor correspondiente vuelve a enviar los paquetes solicitados *a todo el grupo en modo multipunto.* Por el contrario cuando se utiliza un esquema restringido al solicitarse la retransmisión de uno o más paquetes, el sistema retransmisor utiliza algún mecanismo que permita *restringir el alcance de la retransmisión a un cierto número de miembros.*

RMNP utiliza un esquema de ámbito restringido, de modo que cuando un sistema retransmisor realiza el reenvío de determinados paquetes lo hace en modo multipunto pero no al grupo completo sino únicamente a los miembros que son sus descendientes. Adicionalmente, RMNP incorpora las funciones de filtrado de retransmisiones por temporizador y por solicitud explícita, que limitan aún más el alcance de estas retransmisiones.

Muchos de los protocolos de multipunto fiable como es el caso del MTP, el MTP-2 [BOG+94], el XTP o el AFDP [CK96] utilizan esquemas de retransmisión globales. El principal inconveniente de estos esquemas es que hacen un peor uso de los recursos de red que los esquemas restringidos, nótese que cuando un miembro solicita una retransmisión, en la mayoría de los casos esto no significa que todos los miembros del grupo hayan perdido dicho paquete sino que es muy posible que el problema haya sido originado por la congestión de un único encaminador y haya afectado a un número muy pequeño de miembros.

Por otro lado, las retransmisiones restringidas pueden hacerse utilizando distintos mecanismos: (1) hacer las retransmisiones en modo *punto a punto* [KTHB89], ésta no es la solución óptima si el número de miembros al que es preciso reenviar los mismos paquetes es muy elevado; (2) si el número de solicitudes de retransmisión relativas a un mismo paquete supera un determinado umbral, el reenvío se hace en modo multipunto a todo el grupo, y en caso contrario en modo punto a punto; planteamiento seguido en el RMTP y el LBRM [HSC95], y que tiene el inconveniente de que es necesario retardar un cierto tiempo la retransmisión de los paquetes solicitados para verificar si se ha superado o no el mencionado umbral; y (3) realizar las retransmisiones en modo multipunto pero limitando el alcance geográfico. Las formas de limitar dicho alcance son diversas y varían desde utili-

zar el TTL[1] como se hace en el TMTP, a emplear mecanismos de alcance administrativo incorporados en el algoritmo de encaminamiento multipunto, o construir un nuevo grupo multipunto para cada paquete que va a ser retransmitido [BZ93]. Es importante reseñar que este tipo de soluciones esta siendo investigado para incorporar esquemas de ámbito restringido a protocolos de multipunto fiable que operan a nivel de transporte y que dada su complejidad están lejos de estar maduras, y de aquí que muchas propuestas opten por esquemas de retransmisión restringida pero únicamente esbocen posibles soluciones [FJM+95]; sin embargo, en RMNP el coste asociado a realizar retransmisiones restringidas es muy bajo, gracias a ser un protocolo de nivel de red y a utilizar el conocimiento de la red sobre la topología del grupo.

7.3 Agregación y filtrado

Salvo raras excepciones, como es el caso del RMTP, ningún protocolo de transporte de multipunto fiable utiliza funciones análogas a las planteadas en RMNP de agregación y filtrado. Esto se debe a que para realizar estos procesos, es preciso disponer de un árbol de control (árbol RMNP en terminología RMNP) del cual formen parte todos los miembros del grupo. El construir un árbol de control a nivel de transporte (constituido únicamente por sistemas finales), en lugar de hacerlo a nivel de red aprovechando el árbol de distribución, como hace RMNP, plantea los siguientes inconvenientes: (1) dado que a nivel de transporte no se tiene ningún conocimiento sobre la topología de los miembros y la red, el mencionado árbol de control es complejo de construir y mantener (por ejemplo, evitar bucles al producirse cambios en la topología), en cierto modo estos protocolos duplican el esfuerzo nada despreciable ya realizado a nivel de red, para construir y mantener el árbol de distribución; y (2) dado que los miembros son tanto nodos intermedios como nodos hoja en el árbol de control, la entrada/salida en el grupo o el fallo de un miembro puede obligar a una reconstrucción total del árbol de control.

Las funciones de agregación y filtrado, aunque introducen cierto proceso adicional, posibilitan una mejor utilización de los recursos de red y evitan que se produzcan problemas de implosión; consecuentemente, permiten que el protocolo escale adecuadamente con relación al tamaño del grupo. Resultados cuantitativos de los beneficios obtenidos al realizar las funciones de agregación y filtrado ya han sido expuestos en el Capítulo 6 al comparar los resultados obtenidos en la configuración 1 donde no existen SRs y conse-

[1] Este tipo de soluciones son únicamente válidas para protocolos que funcionan sobre IP multipunto, u otro protocolo que disponga de un mecanismo para limitar el tiempo que un datagrama está viajando por la red.

cuentemente no se realizan las funciones de agregación y filtrado, y en la configuración 2 donde se dispone de SRs en los campus cuya función primordial es realizar las funciones de agregación y filtrado.

El mecanismo normalmente utilizado, como es en el caso del SRM, para evitar una implosión de NAKs (en lugar de un proceso de filtrado), es retardar la transmisión de NAKs utilizando temporizadores aleatorios mientras se espera que otro receptor genere un NAK (solicitando los mismos datos), y en el caso de que dicha espera finalice con resultado negativo, enviar el NAK correspondiente en modo multipunto a todo el grupo o restringido a un cierto área. Este planteamiento tiene el inconveniente de retardar el proceso de recuperación de los paquetes perdidos y consecuentemente de aumentar el retardo medio de distribución.

7.4 Nivel de operación

RMNP opera a nivel de red y como ya se ha ido analizando más detalladamente en los puntos anteriores, el hecho de proporcionar fiabilidad a nivel de red tiene como principales ventajas: (1) simplificar la construcción y mantenimiento de un árbol de control, y (2) facilitar la incorporación de un esquema de retransmisión distribuido con alcance restringido.

Esta forma de enfocar el problema proporcionando fiabilidad a nivel de red, ya fue planteada en el FTM, trabajo que no ha tenido continuidad en el tiempo. Sin embargo el FTM difiere del RMNP en los siguientes aspectos:

- El FTM es obligatorio que esté implementado en *todos* los encaminadores de la red, planteando una solución muy poco flexible.

- Está limitado a una arquitectura de encaminamiento multipunto particular, en concreto, el FTM sólo funciona sobre un algoritmo de encaminamiento multipunto concreto propuesto por el mismo autor. Por el contrario, RMNP carece de esta restricción y es capaz de trabajar sobre cualquier arquitectura de encaminamiento multipunto basada en árboles.

- En lugar de plantear un esquema de retransmisiones distribuido, el único sistema retransmisor es la fuente. Y las retransmisiones son siempre globales (multipunto a todo el grupo).

- En lugar de utilizar un esquema de ventana dinámica en distintos sistemas, como hace RMNP, emplea "*ckeckpoints*" periódicos, aproximación que disminuye considerablemente el caudal del protocolo.

La desventaja de proporcionar fiabilidad a nivel de red es que obliga a introducir un nuevo protocolo en determinados encaminadores de la red, con lo cual estos encaminadores al precisar más recursos en CPU y en memoria, serán más costosos o más lentos.

7.5 Esquema de fiabilidad

RMNP utiliza un esquema de fiabilidad orientado al emisor, con delegación de responsabilidades. El hecho de que RMNP resuelva el problema de la fiabilidad utilizando un esquema orientado al emisor, permite lograr un alto grado de fiabilidad, aunque la fuente no conozca la identidad de los miembros del grupo; por el contrario, como se analizó en el Capítulo 4, el seguir una aproximación orientada al receptor tiene como consecuencia que esté bastante limitado el grado de fiabilidad que es posible lograr. La inmensa mayoría de protocolos de multipunto fiable, como por ejemplo el MTP o el SRM, utilizan una aproximación orientada al receptor principalmente con dos finalidades: (1) evitar que la fuente deba conocer la identidad de cada uno de los miembros del grupo, y mantener información de estado asociada a cada uno éstos, y (2) eliminar la probabilidad de que se produzca una implosión de ACKs en la fuente. La forma de resolver estos mismos aspectos en RMNP es mediante: (1) un esquema de delegación de responsabilidades de control de errores, de acuerdo con el árbol RMNP (los sistemas RMNP delegan parte de su responsabilidad en determinados sistemas RMNP que son sus descendientes), y (2) la función de agregación de ACKs.

RMNP prueba que no es cierta la creencia de algunos autores [BOG+94, BZ93] que plantean, que no es posible que un protocolo que utiliza asentimientos positivos sea escalable.

Otro aspecto asociado a la fiabilidad que debe ser destacado, es que la gran mayoría de protocolos de multipunto fiable, como por ejemplo el MTP o el SRM, no incorporan mecanismos para detectar fallos en miembros o particiones en la interred, delegando esta responsabilidad en los niveles superiores. La principal motivación de esta actuación es la complejidad asociada a la incorporación de este tipo de mecanismos en aproximaciones orientadas al receptor.

El hecho de construir el árbol RMNP sobre el árbol de distribución, y la delegación de responsabilidades de control de errores sobre distintos sistemas de dicho árbol RMNP, tiene las múltiples ventajas ya comentadas, pero también hace que sea necesario poner en marcha un procedimiento de reinicio, cuando se produce un cambio en el árbol de distribución que origina una modificación del árbol RMNP; esta limitación impone una cierta sobrecarga en ancho de banda y proceso, aunque debe tenerse en consideración que el árbol de distribución es normalmente estable y dichos cambios serán esporádicos.

7.6 Control de flujo

Para resolver el problema del control de flujo, RMNP plantea un esquema multiventana mientras que otros protocolos orientados al emisor, como el XTP, el RMTP o el protocolo de Crowcroft y Paliwoda [CP88] utilizan una única ventana en la fuente; el utilizar un esquema multiventana tipo RMNP, permite que el protocolo ajuste el tamaño de las distintas ventanas para adaptarse a las necesidades de los miembros accesibles por un mismo interfaz, mientras que cuando sólo existe una única ventana, ésta debe ser compartida por *todos* los miembros del grupo. Por otro lado, muchos de los protocolos de multipunto fiable como es el caso del MTP, el MTP-2 y el AFDP por ser orientados al receptor, únicamente utilizan algún mecanismo de *control del caudal.* Este tipo de mecanismos se basan en restringir (a un cierto número de bytes por segundo), los datos que puede enviar cada fuente en cada unidad de tiempo. Los principales inconvenientes de este tipo de esquemas son: (1) es muy difícil conocer a priori el caudal que necesita la aplicación para poder configurarlo; (2) son menos precisos que los esquemas basados en ventanas, porque la fuente nunca conoce a ciencia cierta si está saturando o no al destino; y (3) obligan a definir explícitamente una velocidad máxima para cada fuente: el cálculo de valores adecuados para estas velocidades es difícil, y normalmente se hace dividiendo el ancho de banda entre el número de fuentes de forma estática, lo cual decrementa la flexibilidad y el caudal alcanzable por estos esquemas.

En este capítulo se ha realizado esencialmente una comparación cualitativa del RMNP frente a otros protocolos de multipunto fiable. Esto tiene por motivación que no existen disponibles datos cuantitativos de simulaciones o implementaciones que puedan ser comparados con los correspondientes del RMNP. Al buscar trabajos que compararan los distintos protocolos de multipunto fiable únicamente se ha localizado un trabajo [PTK94] que presenta una comparación analítica de los protocolos de multipunto fiable orientados al emisor y los orientados al receptor.

Capítulo 8.

CONCLUSIONES Y VÍAS DE INVESTIGACIÓN FUTURAS

En este capítulo se exponen el resumen y las conclusiones principales de esta Tesis Doctoral, y se apuntan unas posibles líneas de continuación de la misma.

El punto focal de este trabajo ha sido el analizar y presentar diversas propuestas al problema de la fiabilidad multipunto en interredes que operan en modo datagrama, considerando las peculiaridades que imponen los hechos de que un grupo esté formado por un número muy elevado de miembros y que la localización geográfica de éstos sea muy dispersa. Así mismo, se ha perseguido una solución que haga un uso óptimo de los recursos de red y que sea aplicable a un amplio rango de aplicaciones, y entre éstas, las sensibles al retardo.

Primeramente, tras estudiar las principales calidades de servicio requeridas normalmente por las aplicaciones multipunto, se ha observado que la fiabilidad es ortogonal a la ordenación y la atomicidad, y consecuentemente no se obtienen ventajas significativas por el hecho de tratarlas juntas diseñando un protocolo que ofrezca fiabilidad y distintos grados de ordenación y atomicidad. Por el contrario, este planteamiento puede dar como resultado un sistema monolítico poco flexible y con dificultades para evolucionar. Consecuentemente, la solución propuesta en este trabajo difiere de la aproximación más tradicional, y separa la fiabilidad de otras garantías de servicio como son la ordenación y la atomicidad. Obviamente, si la aplicación precisa cualidades de servicio adicionales, éstas pueden ser proporcionadas por un protocolo de nivel superior o por la propia aplicación.

Posteriormente, se ha analizado la semántica de la fiabilidad en el entorno multipunto, identificándose diversos aspectos diferenciales de la tradicional comunicación punto a punto, y de acuerdo con esto, se han definido distintos niveles de fiabilidad. Por otro lado, la gran mayoría de las actuales propuestas siguen esquemas orientados al receptor y al mismo tiempo, incorporan el modelo de servicio de Deering; tras el estudio de estos planteamientos se ha concluido que ambos imponen restricciones al grado de fiabilidad proporcionado, limitaciones que son principalmente debidas a que la fuente no conoce la identidad de los miembros del grupo. La solución aquí propuesta utiliza un esquema orientado al emisor, lo cual permite mejorar el nivel de fiabilidad proporcionado, e incorpora el modelo de Deering. El hecho de seguir dicho modelo, permite que el protocolo sea más genérico y consecuentemente aplicable a un mayor número de aplicaciones, mientras que por otro lado si la aplicación requiere que se introduzcan determinadas limitaciones, como que el grupo sea cerrado, esto puede realizarse mediante mecanismos externos. Adicionalmente, para minimizar las restricciones al grado de fiabilidad impuestas por este modelo, se han seguido dos líneas: en primer lugar, incorporar una serie de mecanismos que minimizan la probabilidad de pérdidas, y por otro lado señalizar al usuario del servicio las etapas donde puede decrementarse el nivel de fiabilidad.

La principal contribución de esta Tesis ha sido la presentación de un nuevo protocolo de multipunto fiable para interredes que operan en modo datagrama, éste es el RMNP. El protocolo propuesto opera a *nivel de red*, hecho que no provoca que sea dependiente del algoritmo de encaminamiento multipunto empleado, y sigue una *aproximación orientada al emisor*; así mismo, incluye un *esquema de retransmisiones distribuido*, de modo que la transmisión de los paquetes perdidos se hace desde ciertos sistemas intermedios, y *de ámbito restringido*, es decir, cuando se solicita el reenvío de uno o más paquetes el alcance de la retransmisión está limitado a un cierto número de miembros, incluyendo entre éstos el que hizo la correspondiente solicitud de retransmisión. Adicionalmente, incorpora diversas funciones como las de agregación, filtrado y control intermedio de secuencia que tienen como finalidad disminuir el retardo medio de distribución y optimizar el uso de recursos de red.

El control de flujo en RMNP es realizado mediante un esquema multiventana que permite que el tamaño de cada ventana vaya variando independientemente, ajustándose a las necesidades de los miembros accesibles por un mismo interfaz. Por otro lado, RMNP posibilita que la aplicación especifique un caudal mínimo que debe ser cursado, y en el caso de detectarse que varios miembros no son capaces de cursar dicho caudal, éstos son expulsados del árbol.

Por último, se ha desarrollado un simulador del protocolo RMNP. En los experimentos de simulación realizados se han elegido tres configuraciones significativas y se han ido variando diversos parámetros como el número de miembros o el tamaño de la interred. Dichos experimentos han permitido verificar que el RMNP, escala adecuadamente tanto con relación al tamaño del grupo como con respecto al tamaño de la interred. Adicionalmente, han mostrado que el RMNP ofrece unos valores de retardo de distribución muy satisfactorios y optimiza el uso de los recursos de red, si lo comparamos con un protocolo de las mismas características que sigue una aproximación orientada al emisor tradicional y opera a nivel de transporte.

8.1 Conclusiones

Las principales conclusiones, fruto del trabajo desarrollado, son las siguientes:

- *Es más eficiente resolver el problema de la fiabilidad multipunto a nivel de red que a nivel de transporte.* Son varias las razones que avalan esta afirmación. En primer lugar, las funciones de agregación y filtrado son esenciales para evitar problemas de implosión en la fuente y hacer un buen uso de los recursos de red. Para realizar dichas funciones es imprescindible disponer de un árbol de control, y las tareas de construcción y mantenimiento de dicho árbol se simplifican enormemente a nivel de red, esto es debido al conocimiento que este nivel dispone sobre la topología de la interred. Por otro lado, facilita la implantación de esquemas de retransmisión distribuidos y de ámbito restringido, lo cual es fundamental tanto para disminuir el retardo medio de distribución como para optimizar los recursos de red.

  El RMNP opera a nivel de red pero sin embargo no impone que dicho protocolo deba estar presente en todos los nodos de la interred. Esta es una característica que hace que sea un protocolo muy versátil, y permita configuraciones muy variadas, desde aquellas donde no existe ningún encaminador RMNP en cuyo caso se comporta como un protocolo extremo a extremo, a configuraciones donde todos los sistemas finales e intermedios incorporan la funcionalidad RMNP, obviamente pasando por situaciones intermedias con un número variable de encaminadores RMNP (BRs o SRs) de acuerdo con los requisitos de la aplicación. Adicionalmente, esta gran flexibilidad permite la incorporación de forma escalonada de más encaminadores RMNP en una interred. Naturalmente, dependiendo de cual sea el número, tipo (BRs o SRs) y disposición de la encaminadores RMNP así serán los beneficios obtenidos.

- *Es posible diseñar un protocolo multipunto que ofrezca servicios de fiabilidad siguiendo una aproximación orientada al emisor y que sea escalable.* Desde hace algún tiempo existe la creencia de que aunque las aproximaciones orientadas al emisor son más fiables, sin embargo no es posible seguir este paradigma en el entorno multipunto porque se originan problemas de implosión y las soluciones son poco escalables con relación al número de miembros. Como consecuencia, prácticamente todas las soluciones que tienen en consideración la escalabilidad son orientadas al receptor, con la limitación en el grado de fiabilidad que esto supone. Por el contrario, en el trabajo aquí expuesto se ha mostrado que si a una aproximación orientada al emisor se le incorporan mecanismos de agregación y filtrado, ésta no presenta problemas de implosión y puede exhibir un buen comportamiento con relación a la escalabilidad. Por otro lado, el incorporar estos mecanismos a nivel de red hace que el coste asociado no sea elevado.

- *Para ofrecer un servicio multipunto fiable de calidad es preciso utilizar un mecanismo de retransmisión distribuido y de ámbito restringido.* Estos mecanismos son fundamentales para mejorar el retardo de distribución y ahorrar recursos de red (ancho de banda y proceso). El incorporar estos mecanismos a nivel de red en lugar de hacerlo a nivel de transporte posibilita que se introduzca menos sobrecarga de control en la red, pues se utiliza la información sobre la topología de la red.

  Un mecanismo de retransmisión distribuido obliga a que se guarden copias del mismo paquete en n puntos, utilizando más recursos de memoria. Sin embargo, todas las soluciones que decrementan el retardo medio de distribución imponen cierta sobrecarga de memoria.

- *Utilizar una ventana deslizante como mecanismo de control de flujo de una comunicación multipunto, plantea el problema de que es improbable que el tamaño de dicha ventana se adecúe a las necesidades de todos los miembros.* La anterior afirmación tiene especialmente validez, si como es usual que ocurra, las características de los caminos que conducen a los miembros son muy dispares, o sea, presentan cualidades muy diferentes en retardo de tránsito y ancho de banda. La alternativa elegida para el RMNP ha sido el incorporar un esquema multiventana, de forma que es menor el número de los miembros que comparten la misma ventana. Esta alternativa ha sido considerada la más adecuada frente a no incorporar ningún mecanismo de control de flujo o disponer de una única ventana en la fuente.

8.2 Vías de investigación futuras

A lo largo de esta Tesis se han identificado varias áreas clave, relacionadas con el problema de la fiabilidad multipunto que precisan de una investigación más extensa de lo que ha sido posible en el marco de este trabajo. Adicionalmente, existen varios aspectos que no han sido abordados pero que son importantes para ofrecer un servicio de calidad. Todas estas posibles vías de investigación incluyen, sin orden de preferencia, las siguientes:

- *Incorporar a RMNP mecanismos de reinicio parcial.* En una interred muy extensa, si ocurre un cambio topológico en un determinado punto, es deseable que en lugar de ponerse en marcha un reinicio que afecte a todos los sistemas pertenecientes al árbol RMNP, se resuelva localmente mediante un reinicio parcial de forma que únicamente se vean afectados los sistemas cercanos al punto donde ha ocurrido dicho cambio topológico. Un mecanismo de reinicio parcial también afectará a todo el grupo, al ralentizar el desplazamiento de la ventana general en la fuente, pero sin embargo evitará la retransmisión innecesaria de paquetes en aquellas áreas no afectadas por el problema. Aunque es recomendable refinar el procedimiento de reinicio utilizado por RMNP, debe tenerse en consideración que los árboles de distribución serán normalmente estables, con lo cual los procedimientos de reinicio serán esporádicos.

- *Elección dinámica de qué encaminadores RMNP deben actuar como SRs.* La selección de qué encaminadores deben disponer de capacidad de almacenamiento y retransmisión es un aspecto relevante porque afecta directamente al retardo medio de distribución y a los recursos de red. En la definición del RMNP aquí propuesta, es el administrador de la red quien elige y configura determinados encaminadores para tal propósito. Se considera deseable que se investiguen mecanismos que posibiliten que sea el propio encaminador el que decida, en función de los recursos disponibles y de ciertos parámetros, como por ejemplo, métricas de distancia en la red o el número de miembros descendientes.

- *Continuar investigando sobre los procedimientos de control de flujo en el entorno multipunto.* Es deseable realizar experimentos de simulación del esquema multiventana propuesto en este trabajo, lo cual permitirá refinar dichos mecanismos.

- *Incorporar seguridad al protocolo.* Los aspectos relacionados con la seguridad de una comunicación multipunto han sido raramente abordados hasta la fecha. Sin embargo, la comunicación multipunto tiene un riesgo adicional substancioso, si lo

comparamos con las mismas amenazas en el entorno punto a punto. Esto surge tanto de la ausencia de mecanismos de control de acceso al grupo, como del hecho de que el tráfico multipunto atraviesa potencialmente muchos más enlaces que una única comunicación punto a punto, incrementándose por tanto la posibilidad de un ataque en un enlace.

Apéndice A.

UNIDADES DE DATOS DEL PROTOCOLO

En este apéndice se presentan las unidades de datos del protocolo, exponiéndose el formato y uso de cada uno de los campos de los paquetes RMNP.

Los paquetes RMNP están clasificados por tipo en:

1. *Datos y descubrimiento.* Como su nombre indica, este tipo incluye los paquetes de datos, así como aquellos que son utilizados para el descubrimiento de un valor válido para la MTTU.

2. *Supervisión.* Son aquellos paquetes que se utilizan para supervisar el estado de la comunicación.

3. *Mantenimiento y liberación.* Son aquellos paquetes usados para mantener el árbol RMNP y para liberar una conexión. Por mantenimiento del árbol se entiende, la construcción inicial del árbol en la fase de establecimiento de la conexión, el mantenimiento del árbol en la fase de transferencia (cambios originados por procedimientos de reinicio o por la entrada/salida de miembros) y la eliminación del árbol cuando se libera la conexión.

El formato de los paquetes RMNP depende de su tipo. Todos los paquetes incluyen una misma cabecera, denominada *cabecera común RMNP*. Adicionalmente existe una cabecera propia para cada tipo, de modo que cada paquete en concreto, incluye la cabecera común y una segunda cabecera específica de su tipo.

Seguidamente se detalla el formato de la cabecera común, para posteriormente, ir exponiendo para cada uno de los tipos, tanto el formato de su cabecera específica como los distintos paquetes propios del tipo.

A.1 Formato de la cabecera común

Todos los paquetes RMNP comienzan con la siguiente cabecera:

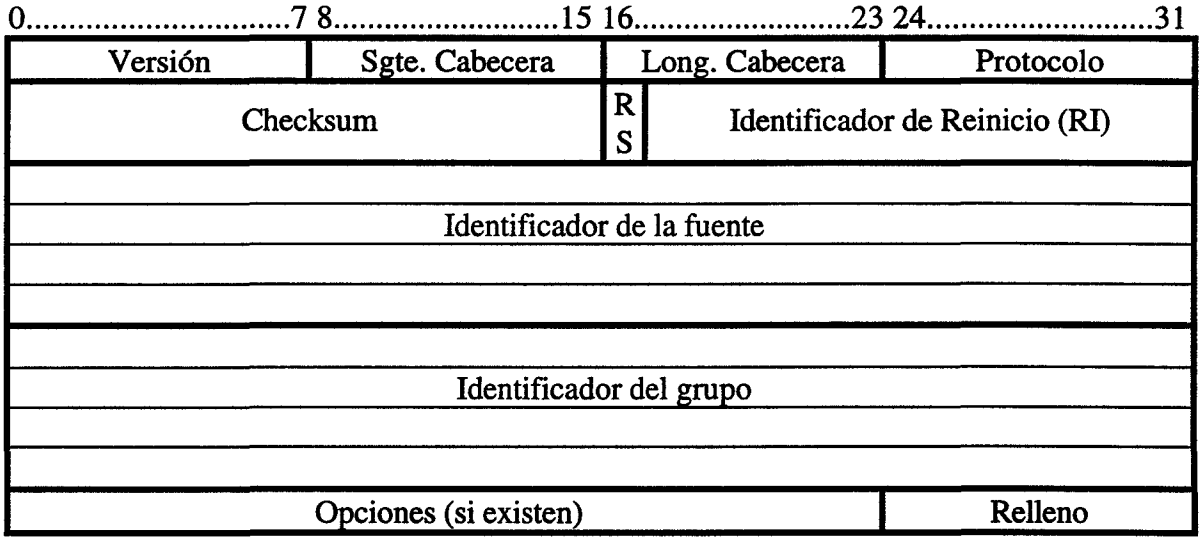


Figura A.1: Cabecera común RMNP

El significado de cada uno de los campos de la cabecera común, es el siguiente:

- ***Versión***: especifica el número de versión del protocolo. Este documento describe la versión 1 del RMNP. Cualquier paquete recibido, con un número de versión inválido será rechazado.

- ***Sgte. Cabecera***: indica cual es el tipo de la cabecera que sigue a la cabecera común (debe observarse que también indica de qué tipo es el paquete). Los distintos valores que puede tomar este campo, así como su significado, se muestran en la siguiente tabla:

Valor	*Semántica*
1	Datos y descubrimiento
2	Supervisión
3	Mantenimiento y liberación

- ***Long. Cabecera:*** especifica la longitud de la cabecera en múltiplos de 32 bits.

- ***Protocolo:*** identificador del protocolo de nivel superior.

- ***Checksum:*** complemento a uno de la suma complemento a uno de todas las palabras de 16 bits del paquete, incluyendo cabeceras y datos de usuario. El propósito de este campo es determinar si el paquete ha llegado libre de errores. Para calcular y verificar el checksum, el propio campo checksum se considera relleno de ceros. Si el paquete RMNP completo (cabeceras y datos) no es un múltiplo de palabras de 16 bits, al realizar el cálculo y verificación del checksum se añade un octeto a ceros al final.

- ***RS***: flag de estado del reinicio. Si este flag está activado, significa que el procedimiento de reinicio identificado en el campo RI está activo, mientras que en caso contrario significará que dicho procedimiento ya ha concluido.

- ***Identificador de Reinicio (RI):*** identificador del procedimiento de reinicio en proceso. Si este campo está a cero significa que no hay ningún procedimiento de reinicio activo.

- ***Identificador de la fuente:*** dirección de la fuente (se asume un esquema de direccionamiento de 128 bits).

- ***Identificador del grupo:*** dirección del grupo multipunto (se asume un esquema de direccionamiento de 128 bits).

- ***Opciones***: campo concebido para proporcionar futuras mejoras al RMNP. De manera similar al TCP, cada especificación de una opción tiene el formato tipo-longitud-valor. En este formato, el campo tipo es un octeto que indica de que tipo es la opción; el campo longitud es un octeto que indica la longitud en bytes del campo valor; y el campo valor son uno o varios octetos conteniendo los datos específicos de la opción.

- ***Relleno***: este campo únicamente será necesario cuando se especifiquen una o más opciones, y tiene como finalidad que la cabecera sea un múltiplo de 32 bits.

A.2 Tipo datos y descubrimiento

A.2.1 Paquetes del tipo datos y descubrimiento

A continuación se detallan los distintos paquetes que pertenecen a este tipo, analizando brevemente sus principales funciones:

- ***DAT.*** Estos paquetes son utilizados por la fuente para distribuir datos de usuario a los miembros. Estos paquetes se propagan siguiendo el árbol de distribución.

- ***MTTUD.*** Estos paquetes son utilizados por la fuente, junto a los MTTUD_ACKs, para descubrir un valor válido para la MTTU, de acuerdo con el algoritmo visto en la sección 5.4.4. Estos paquetes se propagan siguiendo el árbol de distribución desde la fuente hacia los miembros.

- ***MTTUD_ACK.*** Estos paquetes son la confirmación a los MTTUDs y como se ha comentado, son utilizados por la fuente para descubrir un valor válido para la MTTU. Estos paquetes son generados por los miembros y se propagan por el árbol RMNP desde éstos hacia la fuente, y en los encaminadores se les va aplicando un proceso de agregación.

A.2.2 Formato de la cabecera de datos y descubrimiento

Todos los paquetes del tipo datos y descubrimiento incluyen la siguiente cabecera:

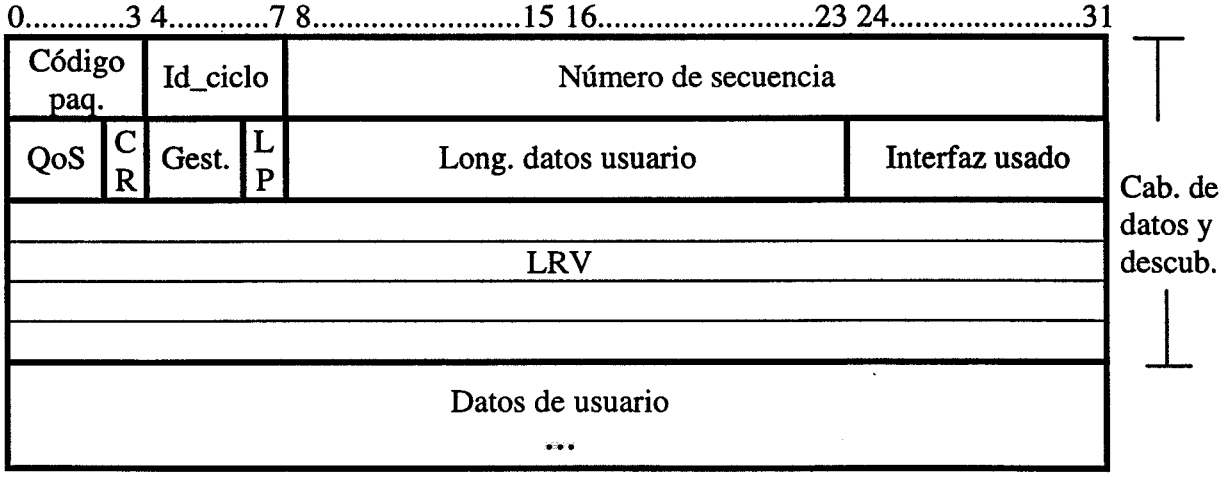


Figura A.2: Cabecera de datos y descubrimiento

El significado de cada uno de los campos de esta cabecera, es el siguiente:

- ***Código paq.***: indica el subtipo del paquete, de acuerdo con los valores mostrados en la siguiente tabla:

Valor	*Semántica*
1	DAT
2	MTTUD
3	MTTUD_ACK

- ***Id_ciclo:*** identificador del ciclo de descubrimiento. Identifica el ciclo del *algoritmo de cálculo de un valor válido de MTTU,* en el cual ha sido generado el paquete MTTUD o MTTUD_ACK. Este identificador es incluido por la fuente al generar un MTTUD y permite que el resto de los sistemas puedan asociar de forma correcta cada MTTUD_ACK con su correspondiente MTTUD.

- ***Número de secuencia (SN):*** número de secuencia del paquete de datos. El primer paquete de datos que envía la fuente tiene como número de secuencia un 1. Ya que los números de secuencia tienen 24 bits de longitud, su rango válido va desde 0 a 2^{24}-1; como este rango es finito, toda la aritmética y comparación de estos números debe ser módulo 2^{24}.

- ***QoS***: especifica el QoS deseado para la conexión RMNP. La definición actual del RMNP sólo contempla una QoS de multipunto fiable según el modelo de Deering, consecuentemente, este campo tiene un único valor válido:

Valor	*QoS deseado*
1	Multipunto fiable según el modelo de Deering

- ***CR***: flag solicito conexión. Este flag permite que la fuente pueda indicar su deseo de establecer una conexión RMNP. El bit CR va activado en todos los paquetes excepto el último, de la primera ventana de transmisión.

- ***Gest.:*** flags de gestión de ventanas. Estos tres flags permiten que la fuente indique a los SRs si está permitido y en que grado, reducir el tamaño de las ventanas. En concreto, la fuente utiliza los siguientes flags:

  - **Flag de Prohibido reducir (FD)** (bit 4): la fuente activa este flag para prohibir que el tamaño de las ventanas sea decrementado.

- ♦ **Flag de Incrementar ventana (IW)** (bit 5): si este flag está activado significa que con el tamaño actual de las ventanas es imposible cursar el caudal mínimo y consecuentemente, el tamaño de todas las ventanas debe ser incrementado.

- ♦ **Flag de Reducciones provisionales (PD)** (bit 6): si este flag está activado, la fuente permite que se decremente el tamaño de las ventanas pero de forma provisional.

❒ ***LP:*** flag de último paquete. Este flag permite que la fuente indique cual es el último paquete generado, al segmentar cada uno de los mensajes recibidos del nivel superior.

❒ ***Long. datos usuario:*** longitud en octetos del campo de datos de usuario que sigue a la cabecera de datos y descubrimiento. Esta longitud será cero cuando se mande un paquete de datos vacío.

❒ ***Interfaz usado (I):*** identificador local del interfaz que fue usado para enviar el paquete, por el último sistema RMNP que ha visitado dicho paquete.

❒ ***LRV:*** identidad del último sistema RMNP que ha visitado el paquete (se asume un esquema de direccionamiento de 128 bits).

Todos los paquetes del tipo datos y descubrimiento incluyen, después de las cabeceras (común y específica), un campo de datos de usuario; en el caso de los paquetes DAT este campo llevará realmente datos de usuario y en los paquetes MTTUD irá a ceros. Debe observarse que en los paquetes DAT no tiene significado el campo Id_ciclo; y del mismo modo, en los paquetes MTTUD y MTTUD_ACK carecen de significado los campos: SN, QoS y LP. Obviamente los paquetes MTTUD_ACK son paquetes vacíos (tienen como longitud del campo datos de usuario un cero).

A.3 Tipo supervisión

A.3.2 Paquetes del tipo supervisión

A continuación se detallan los distintos paquetes que pertenecen a este tipo, analizando brevemente sus principales funciones:

- ***ACK***. Estos paquetes son utilizados por los miembros para confirmar la recepción correcta de paquetes DAT. Los paquetes ACK se propagan por el árbol RMNP desde los miembros hacia la fuente, y en los encaminadores se les aplica un proceso de agregación.

- ***NAK***. Estos paquetes son utilizados por distintos sistemas RMNP (miembros y encaminadores), para solicitar el reenvío de ciertos paquetes de datos al sistema retransmisor más cercano (SR o fuente). Los paquetes NAK se propagan por el árbol RMNP de abajo-arriba, y en los encaminadores se les aplica un proceso de filtrado.

A.3.2 Formato de la cabecera de supervisión

Todos los paquetes de supervisión incluyen la siguiente cabecera:

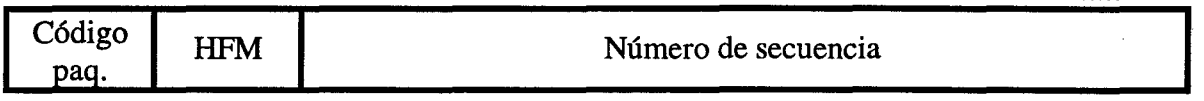


Figura A.3: Cabecera de supervisión

El significado de cada uno de los campos de la cabecera de supervisión, es el siguiente:

- ***Código paq.***: indica el subtipo del paquete, de acuerdo con los valores mostrados en la siguiente tabla:

Valor	*Semántica*
1	ACK
2	NAK

- ***HFM:*** número de saltos RMNP al miembro más alejado. La motivación y utilidad de este campo se expuso al analizar el ajuste de MAX_RET_BCR_h_i.

❒ ***Número de Secuencia (SN):*** la semántica de este campo varía según el paquete sea un ACK o un NAK. En cada uno de los casos, un valor de *n* en este campo significa lo siguiente: (1) si el paquete es un ACK significa que el sistema, que ha generado este ACK, ha recibido correctamente hasta el paquete de datos con número de secuencia *n* incluido éste; (2) si el paquete es un NAK significa que el sistema, que ha generado este NAK, solicita la retransmisión de los paquetes de datos pendientes de confirmación empezando con el que tiene número de secuencia *n*.

A.4 Tipo mantenimiento y liberación

A.4.1 Paquetes del tipo mantenimiento y liberación

A continuación se detallan los distintos paquetes que pertenecen a este tipo, analizando brevemente sus principales funciones:

❒ *TS*. Estos paquetes de *estado del árbol* tienen como finalidad el mantenimiento del árbol RMNP y son utilizados por distintos sistemas RMNP (miembros y encaminadores) con diversas finalidades, que se verán a continuación:

♦ Son utilizados por un sistema para avisar *a su padre* RMNP de lo siguiente:

- Soy tu nuevo hijo y deseo entrar a formar parte del árbol RMNP (activando el flag NC explicado en la siguiente sección).

- Deseo dejar de formar parte del árbol RMNP (activando el flag ML explicado en la siguiente sección).

- Se ha iniciado un procedimiento de reinicio y es preciso comprobar que continúo siendo tu hijo (activando el flag TC explicado en la siguiente sección).

- Ha cambiado el interfaz que utilizas para mandarme paquetes y es necesario actualizarlo (activando el flag NI explicado en la siguiente sección).

En cualquiera de estos casos, el hijo se mantendrá a la espera de una contestación de su padre (un paquete TS_ACK con el flag correspondiente activado).

- Son utilizados por un sistema (encaminador o miembro) para avisar *a la fuente* de que se ha detectado un cambio en el árbol RMNP, y como consecuencia ésta debería poner en marcha un reinicio (activando el flag RN explicado en la siguiente sección). El sistema se mantendrá a la espera de detectar que la fuente, como consecuencia este aviso, ha puesto en marcha un procedimiento de reinicio.

❒ ***TS_ACK***. Estos paquetes son utilizados por los padres RMNP para confirmar a sus hijos la recepción de un paquete TS, y para indicarles cierta información de estado, tal como:

- Cual es el último paquete que ellos mismos han confirmado a su propio padre. Esta información permite a un nuevo hijo, que se unió al grupo en la mitad de la fase de transferencia, conocer a partir de qué paquete debe esperar.

- Cual es el identificador del último reinicio conocido. Dicho identificador va contenido en el campo Identificador de reinicio de la cabecera común, que como puede observarse, en estos paquetes TS_ACK cambia ligeramente su semántica.

- Estado del último reinicio conocido.

❒ ***LIB***. Estos paquetes de liberación son usados para liberar parcial o totalmente una conexión RMNP. Se entiende por liberación total cuando el procedimiento de liberación es iniciado por la fuente, mientras que una liberación parcial es iniciada por un encaminador y tiene como resultado que uno o más miembros dejarán de pertenecer al árbol RMNP o lo que es lo mismo, dejarán de participar en la conexión RMNP (nótese que una conexión RMNP es multiparte y en ella participan una fuente y *n* miembros). De acuerdo con esto, los paquetes LIB se utilizan en las siguientes situaciones:

- La fuente utiliza los paquetes LIB para liberar de forma ordenada una conexión RMNP (activando el flag OR explicado en la siguiente sección), y para abortar una conexión RMNP (activando el flag AB explicado en la siguiente sección).

- Los encaminadores utilizan los paquetes LIB para expulsar del árbol RMNP a uno o más hijos y sus correspondientes descendientes (liberar parcialmente una conexión). Así mismo, son usados por dichos encaminadores para avisar a la fuente del hecho anterior (activando el flag INF explicado en la siguiente sección).

❒ ***LIB_ACK.*** Todo proceso de liberación ordenada de una conexión RMNP es confirmado, y esta confirmación la hacen los miembros utilizando paquetes LIB_ACK. Estos paquetes se propagan por el árbol RMNP desde los miembros hacia la fuente, y en los encaminadores se les aplica un proceso de agregación.

A.4.2 Formato de la cabecera de mantenimiento y liberación

Todos los paquetes de mantenimiento y liberación incluyen la siguiente cabecera:

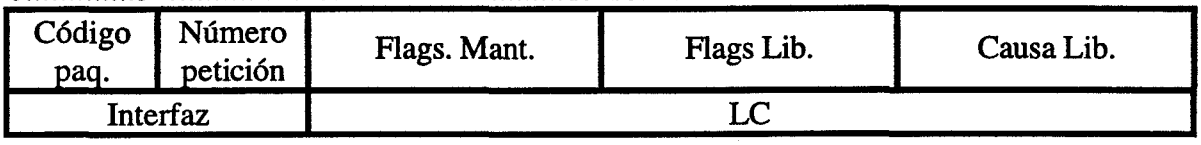


Figura A.4: Cabecera de mantenimiento y liberación

El significado de cada uno de los campos de esta cabecera es el siguiente:

❒ ***Código paq.:*** indica el subtipo del paquete, de acuerdo con los valores mostrados en la siguiente tabla:

Valor	*Semántica*
1	TS
2	TS_ACK
3	LIB
4	LIB_ACK

❒ ***Número petición:*** identificador que permite asociar cada paquete TS con su correspondiente confirmación TS_ACK.

❒ ***Flags Mant.:*** flags asociados con el mantenimiento del árbol RMNP:

- *NC* (bit 8): flag de nuevo hijo. Activando este flag en un paquete TS, un sistema RMNP solicita a su padre entrar a formar parte del árbol.

- *ML* (bit 9): flag de salida de un miembro. Activando este flag en un paquete TS, un sistema solicita a su padre dejar el árbol RMNP.

- ***TC*** (bit 10): flag de comprobación del árbol. Activando este flag en un paquete TS, un sistema solicita a su padre que compruebe, si a pesar del cambio en el árbol RMNP su relación padre-hijo continúa inalterable. Este tipo de comprobaciones se hacen cuando se pone en marcha un procedimiento de reinicio.

- ***NI*** (bit 11): flag de nuevo interfaz. Activando este flag en un paquete TS, un sistema informa a su padre de que ha cambiado el interfaz por el cual está accesible y le solicita que actualice su TIB.

- ***RN*** (bit 12): flag de reinicio necesitado. Activando este flag en un paquete TS, un determinado sistema (encaminador o miembro) avisa a la fuente de que se ha producido un cambio en el árbol RMNP, y es preciso poner en marcha un procedimiento de reinicio.

❒ ***Flags Lib.:*** flags asociados con la liberación parcial o total de una conexión RMNP:

- ***OR*** (bit 16): flag de liberación ordenada. Este flag indica que con el paquete LIB se está produciendo una liberación ordenada de la conexión.

- ***AB*** (bit 17): flag de aborto. Este flag indica que con el paquete LIB se está llevando a cabo un aborto de la conexión.

- ***INF*** (bit 18): flag de información de liberación. Si un paquete LIB lleva activado este flag indicará que el sistema que generó el paquete (un encaminador) ha liberado parcialmente la conexión (expulsando a uno o más miembros del árbol) y desea informar del hecho a la fuente (destino del paquete).

❒ ***Causa Lib.***: especifica la causa del aborto de la conexión. Los valores válidos de este campo son los mostrados en la siguiente tabla:

Valor	*Semántica*
1	Se ha alcanzado el número máximo de retransmisiones permitidas.
2	Imposibilidad de cursar el caudal mínimo exigido.
3	Se ha alcanzado MAX_TWF.
4	Se ha alcanzado MAX_TWS.

- ***Interfaz (I):*** identificador del interfaz usado por un padre para distribuir paquetes a su hijo. Utilizando este campo en un paquete TS, un hijo informa a su padre de cual es el identificador del interfaz que dicho padre emplea para enviarle paquetes.

- ***LC:*** Último paquete confirmado. Utilizando este campo en un paquete TS_ACK, un padre informa a su hijo de cual es el último paquete que el mismo a confirmado a su propio padre.

Apéndice B.

PSEUDOCÓDIGO DE LOS ENCAMINADORES RMNP

En este apéndice se detalla la operación del RMNP utilizando pseudocódigo, y en concreto se muestra el comportamiento de los encaminadores RMNP (BRs y SRs). El hecho de elegir los encaminadores ha sido motivado porque éstos se consideran los sistemas más representativos, ya que incluyen funciones específicas como las de agregación y filtrado, y por otro lado, comparten funciones con la fuente y los miembros; por ejemplo, al igual que la fuente son sistemas retransmisiones y al igual que los miembros realizan control de secuencia.

El pseudocódigo se encuentra organizado en base a un conjunto de estados en los que ocurren una serie de sucesos. La ocurrencia de un suceso provoca unas acciones y puede causar o no la transición a otro estado distinto. En todos los estados existe un cuerpo principal que está ejecutándose mientras no se dé ningún suceso, cuando ocurre un suceso se salta en forma de subrutina, se ejecuta el código asociado a dicho suceso; y posteriormente, o bien, se vuelve al punto donde se detuvo la ejecución en el cuerpo principal, mediante la instrucción RETURN, si el suceso ocurrido permite quedarse en el mismo estado, o bien, se transita hacia otro estado, mediante la instrucción ESTADO cuyo argumento es precisamente el nuevo estado al que se pasa. El estado Retransmisión permite adicionalmente que cada interfaz hijo tenga asociado un subestado (retransmisión por temporizador o por solicitud explícita); la transición de un interfaz a un subestado se hace mediante la instrucción SUBESTADO cuyo argumento es el nombre del subestado al que se pasa.

Las referencias a los campos de las unidades de datos del protocolo siguen la siguiente nomenclatura: paquete.campo. De este modo el campo SN del paquete DAT, se referencia como DAT.SN.

Con la única finalidad hacer un pseudocódigo más legible, existen una serie de módulos que son utilizables en todos los estados; para llamar a uno de éstos se utiliza la instrucción CALL_MOD que tiene como argumento el nombre del módulo correspondiente. Adicionalmente, algunos estados tienen un conjunto de procedimientos propios de un suceso, la instrucción utilizada para llamar a dichos procedimientos es CALL siendo el argumento el nombre del procedimiento correspondiente.

MÓDULOS UTILIZABLES EN TODOS LOS ESTADOS

□ *Módulo* Comprueba_cambio_árbol

```
{En este módulo se verifica si ha habido un cambio en el árbol RMNP, y con este fin se comprueba
si ha variado la identidad del padre. En el caso de que haya cambiado el padre se manda un paquete
TS a la fuente indicándole que debe poner en marcha un proceso de reinicio}
        IF NOT (TIB.padre = DAT.LRV) THEN
            IF NOT (Pendiente_reinicio) THEN
                Pendiente_reinicio = true
                TS.RN = Activado                    {Flag de reinicio necesitado activado}
                Incrementa_Último_RI    {Actualiza el identificador del último reinicio}
                TS.RI = Último_RI
                Envía_TS(fuente)
                Calcula_Tws
                Activa_Tws                  {Activa el temporizador de espera TS_ACK}
            ENDIF
            Descarta_paquete
            RETURN
        ENDIF
    END_módulo
```

□ *Módulo* Comprueba_reinicio

```
{En este módulo se comprueba si la fuente ha puesto en marcha un procedimiento de reinicio}
        IF (DAT.RS = Activado) and (DAT.RI ≠ 0) THEN
```

```
                    {Comprueba si el paquete recibido pertenece a un reinicio anterior}
        IF (DAT.RI < Último_RI) THEN
            Descarta_paquete
            RETURN
        ENDIF
          {Si se había solicitado un reinicio a la fuente, detiene el temporizador de esperando a la fuente}
        IF (está activado T_WS) and (Pendiente_reinicio) THEN
            RESET(T_WS)
        ENDIF
        IF (es un SR) THEN Libera_buffers ENDIF
        Último_RI = DAT.RI                    {Actualiza información_último_reinicio}
        Actualiza_INF_de_TIB(padre)
        TS.RI = Último_RI                     {Construye un paquete TS para su padre}
        TS.TC = Activado                      {Flag de comprobación del árbol activado}
        Envía_TS(padre)                       {Solicita a su padre la actualización de la TIB}
        Calcula_T_WF
        Activa_T_WF                           {Activa el temporizador de espera TS_ACK}
                                              {Distribuye el paquete por todos los interfaces}
        DAT.LRV = Identificador_propio
        FOR i IN n_interfaces DO
            DAT.I = i
            Difunde_paquete_X_interfaz[i]
        END_FOR
                                              {Comprueba si es un reinicio o un reinicio suave}
        IF (DAT.dat_usuario = vacío) THEN
            Reinicio_suave = true
        ELSE
            Borra_INF_Est(padre)
            FOR i IN n_hijos DO
                Borra_INF_Est[i]
            END_FOR
        ENDIF
            ESTADO (Reinicio)
    ENDIF
END_módulo
```

□ *Módulo* Comprueba_interfaz

```
{Comprueba si ha variado el interfaz que utiliza su padre para mandarle datos}
    IF NOT (TIB.padre.interface = DAT.I) THEN
```

```
        Actualiza_INF_de_TIB(padre)
        TS.NI = Activado                          {Flag de nuevo interfaz activado}
        Indica_nuevo_interfaz_en_TS
        Envía_TS(padre)
        Calcula_T_WF
        Activa_T_WF                    {Activa el temporizador de espera TS_ACK}
    ENDIF
END_módulo
```

- *Módulo* Comprueba_MAX_TWF

```
{Comprueba si el número de paquetes enviados al padre, sin que éste haya contestado, ha alcanzado el valor del parámetro de configuración MAX_TWF, en cuyo caso hace una liberación parcial y avisa de este hecho a la fuente}
    IF (num_retransmisiones_a_padre = MAX_TWF) THEN
        FOR i IN n_interfaces DO
            Baja_INF_de_TIB(hijos_interfaz)
            Baja_INF_Est(hijos_interfaz)
            LIB.AB = Activado                          {liberación parcial}
            Envía_LIB(i)
        END_FOR
        LIB.INF = Activado
        Envía_LIB(fuente)              {Informa a la fuente de la liberación parcial}
        Calcula_T_WS
        Activa_T_WS
        IF (es un SR) THEN Libera_buffers ENDIF
        ESTADO (Avisando Liberación Parcial)
    ENDIF
END_módulo
```

- *Módulo* Comprueba_MAX_TWS

```
{Comprueba si el número de paquetes enviados a la fuente, sin que ésta haya contestado, ha alcanzado el valor del parámetro de configuración MAX_TWS, en cuyo caso pone en marcha una liberación parcial}
    IF (num_retransmisiones_a_fuente = MAX_TWS) THEN
        FOR i IN n_interfaces DO
            LIB.AB = Activado
            Envía_LIB(i)
        END_DO
        Libera_recursos
        ESTADO (Fin)
```

```
    ENDIF
END_módulo
```

Estado Distribución Normal

```
BUCLE_INFINITO
    WHILE NOT(existe suceso) DO
        CALL Proceso_ocioso
    END_WHILE
    CASE suceso IN
        Recibe_DAT:          GOSUB(Recibe_DAT)
        Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
        Recibe_MTTUD_ACK:    GOSUB(Recibe_MTTUD_ACK)
        Recibe_ACK:          GOSUB(Recibe_ACK)
        Recibe_NAK:          GOSUB(Recibe_NAK)
        Recibe_TS:           GOSUB(Recibe_TS)
        Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
        Recibe_LIB:          GOSUB(Recibe_LIB)
        Recibe_LIB_ACK:      GOSUB(Recibe_LIB_ACK)
        Expira_T_RET:        GOSUB(Expira_T_RET)
        Expira_T_OS:         GOSUB(Expira_T_OS)
        Expira_T_LIB:        GOSUB(Expira_T_LIB)
        Expira_T_WF:         GOSUB(Expira_T_WF)
        Expira_T_WS:         GOSUB(Expira_T_WS)
    ESAC
FIN_BUCLE
```

Suceso: ***Recibe_DAT***

{Recibe desde su padre un DAT_n}

{Actualiza el temporizador de liberación por silencio de la fuente}

Actualiza (T_{LIB} , MAX_VAL_TLIB)

{Comprueba si se ha iniciado un proceso de reinicio}

CALL_MOD (Comprueba_reinicio)

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}

CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que su padre le envía paquetes de datos. Si es así avisará a éste}

CALL_MOD (Comprueba_interfaz)

{Procesa el paquete recibido según éste sea: el paquete esperado en secuencia, un paquete fuera de secuencia, un duplicado no confirmado previamente o un duplicado ya confirmado. Antes de realizar esta clasificación comprueba si el paquete recibido está dentro de la ventana y para ello utiliza el procedimiento Dentro_Ventana_R. Con este fin necesita la siguiente información de estado:
- Esperado: es el siguiente paquete en secuencia
- Ultimo_confirmado: último paquete confirmado a su padre}

```
IF (Dentro_Ventana_R) THEN
                                                    {Recibe paquete en secuencia}
   IF (DAT.SN = Esperado) THEN
        Actualiza_contador_paquete_esperado
        CALL Distribuye_paquetes
                                                    {Recibe paquete fuera de secuencia}
   ELSE_IF (DAT.SN > Esperado) THEN
              CALL Fuera_de_secuencia
                                                    {Recibe un duplicado sin confirmar}
         ELSE_IF (DAT.SN < Esperado) and (DAT.SN > Ultimo_confirmado) THEN
                    CALL Duplicado_NO_confirmado
                                                    {Recibe un duplicado ya confirmado}
                ELSE {DAT.SN ≤ Ultimo_confirmado}
                          CALL Duplicado_YA_confirmado
                ENDIF
         ENDIF
   ENDIF
ELSE {paquete Fuera de la Ventana}
   Error (fuera ventana)
ENDIF
RETURN
```

PROCEDIMIENTOS DEL SUCESO Recibe_DAT

□ *Proceso* Dentro_Ventana_R

{ENS es el Espacio de Números de Secuencia}
{W_G es el tamaño de la ventana general}

```
IF (DAT.SN > mod_ENS(Ultimo_confirmado + W_G) and
                                   (DAT.SN < mod_ENS(Esperado - W_G))  THEN
    Dentro_Ventana_R = FALSE
ELSE
    Dentro_Ventana_R = TRUE
```

```
    ENDIF
END_Proceso
```

□ *Proceso* Distribuye_paquetes

{Envía el paquete recibido a los hijos; después comprueba si existen paquetes fuera de secuencia almacenados temporalmente y si el paquete recibido completa la secuencia; en el caso de ser así, también distribuye dichos paquetes fuera de secuencia}

```
    CALL Envía_paquete_hijos
    WHILE (activo T_OS) and (PILA_paq_fuera_secuencia.NS = esperado) DO
        DAT = GET (PILA_paq_fuera_secuencia)
        Actualiza_contador_paquete_esperado
        IF (PILA_paq_fuera_secuencia está vacía) THEN RESET (T_OS) ENDIF
        CALL Envía_paquete_hijos
    END_WHILE
END_Proceso
```

□ *Proceso* Envía_paquete_hijos

{El proceso de distribución a los hijos depende de si el encaminador es un BR o un SR}

```
IF (es un BR) THEN
    DAT.LRV = Identificador_propio
    FOR i IN n_interfaces DO
        DAT.I = i
        Difunde_paquete_X_interfaz[i]
        Incrementa_NUM_RET[n,i]
    END_FOR
ELSE  {El encaminador es un SR}
    DAT.LRV = Identificador_propio
    Almacena el paquete
    FOR i IN n_interfaces DO
        DAT.I = i
            {Comprueba el estado de la ventana del interfaz "i", si se ha alcanzado el tamaño máximo de dicha ventana, retarda el envío del paquete por dicho interfaz mientras continúa distribuyéndolo por otros interfaces}
        WHILE (W_INT[i] está saturada) DO
            Retardo_para_interfaz[i]
        END_WHILE
        Difunde_paquete_X_interfaz[i]
        Incrementa_NUM_RET[n,i]
        Actualiza_límite_superior_W_INT[i]        {Actualiza límite de la ventana}
        Calcula_T_RET[n,i]                        {Realiza el cálculo del temporizador}
```

```
            Activa_T_RET[n,i]
        END_FOR
    ENDIF
    END_módulo
```

□ *Proceso* Fuera_de_secuencia

{Almacena el paquete en la pila de fuera de secuencia, ordenada de menor a mayor, y activa el temporizador de fuera de secuencia, si no está activado}

```
        PUT (PILA_paq_fuera_secuencia, DAT)
        ORDENA (PILA_paq_fuera_secuencia)
        IF NOT (T_OS está activo) THEN Activa_T_OS
    END_Proceso
```

□ *Proceso* Duplicado_NO_confirmado

{En el caso de ser un SR descarta el paquete. Si es un BR distribuye el paquete aplicando la función de filtrado de retransmisiones por temporizador. Para decidir se debe filtrar o no utiliza la variable NUM_RET[n,i] que almacena el número de retransmisiones del paquete "n" por el interfaz "i". Posteriormente, intenta detectar posibles particiones en la interred, para esto utiliza como variable auxiliar NUM_RET_TEMP[n,i] que almacena el número de retransmisiones del paquete "n" por el interfaz "i" originadas por el vencimiento de un temporizador}

```
        IF (es un SR) THEN Descarta_paquete
        ELSE                                        {El encaminador es un BR}
            DAT.LRV = Identificador_propio
            FOR i IN n_interfaces DO
              DAT.I = i
                                    {Aplica Filtrado de Retransmisiones por Temporizador}
              IF (NUM_RET[n,i] ≤ MAX_FIL) OR (NUM_RET[n,i] es par)
                  THEN                  {filtra}
                        IF NOT (paquete_confirmado_todos_hijos_ interfaz[i]) THEN
                              Difunde_paquete_X_interfaz[i]
                              Incrementa_NUM_RET[n,i]
                              Incrementa_ NUM_RET_TEMP[n,i]
                        ENDIF
                  ELSE                  {no filtra}
                        Difunde_paquete_X_interfaz[i]
                        Incrementa_NUM_RET[n,i]
                        Incrementa_ NUM_RET_TEMP[n,i]
              ENDIF
                              {Detecta posibles anomalías como una partición en la interred}
              FOR h_i IN n_hijos_interfaz[i] DO
```

```
                    Calcula_ MAX_RET_BCR_hi
                    IF (NUM_RET_TEMP[n,i] > MAX_RET_BCR_hi) and
                                        (hi no ha confirmado el paquete recibido) THEN
                        Baja_INF_de_TIB(hi)
                        Baja_INF_Est(hi)
                        LIB.AB = Activado      {Informa al hijo de la liberación parcial}
                        Envía_LIB(hi)
                        Liberación_parcial_en_curso = true
                    ENDIF
                  END_FOR
                END_FOR
                  IF Liberación_parcial_en_curso THEN
                        LIB.INF = Activado          {Avisa a la fuente de la liberación parcial}
                        Envía_LIB(fuente)
                        Calcula_Tws
                        Activa_Tws
                        IF (número_hijos = 0 ) THEN
                            ESTADO (Avisando Liberación Parcial)
                             {Comprueba si el resto de los hijos ya le habían confirmado el
                             paquete recibido}
                          ELSE_IF NOT(existe hijo por confirmar paquete n) THEN
                            ACK.SN=n
                            Envía_ACK(padre)
                            Actualiza_INF_Est(padre)
                          ENDIF
                        ENDIF
                  ENDIF
            ENDIF
        END_Proceso
```

□ *Proceso* Duplicado_YA_confirmado

```
    Descarta_paquete
    ACK.SN = Ultimo_confirmado
    Envía_ACK(padre)
END_Proceso
```

Suceso: ***Recibe_MTTUD***

{Lo envía a sus hijos a través del árbol de distribución}

```
                    {Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (TLIB , MAX_VAL_TLIB)
```

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}

CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que le envía el padre los paquetes de datos. Si es así avisará a su padre}

CALL_MOD (Comprueba_interfaz)

{Distribuye el paquete por todos los interfaces}

MTTUD.LRV = Identificador_propio
FOR i IN n_interfaces DO
  MTTU.I = i
  Difunde_paquete_X_interfaz[i]
END_FOR
RETURN

Suceso: ***Recibe_MTTUD_ACK***

{Realiza el proceso de agregación de MTTUD}

Actualiza_INF_Est(hijo)
IF NOT(existe hijo por confirmar MTTUD) THEN
  Envía_MTTUD_ACK {Envía confirmación del paquete MTTUD al padre}
ENDIF
RETURN

Suceso: ***Recibe_ACK***

{Recibe desde un hijo asociado al interfaz "i" un ACK_n. Actualiza la información de estado asociada al hijo que le envió el asentimiento, y si procede realiza un proceso de agregación de ACKs}

Actualiza_INF_Est(hijo)

{En el caso de un SR, si todos los hijos accesibles por el interfaz "i" han confirmado el paquete "m" (siendo "m" ≤ "n"), detiene el temporizador de retransmisión asociado "i" y actualiza el límite de la ventana}

IF (es un SR) and (todo hijo[i] confirmó m) and (T_{RET}[m,i] está activo) THEN
  RESET (T_{RET}[m,i])
  Actualiza_límite_inferior_W_{INT}[i] {Actualiza límite de la ventana}
ENDIF

{Proceso de agregación de ACKs. Comprueba el paquete mínimo "m" ≤ "n" confirmado por todos los hijos, y envía un ACK al padre}

IF NOT(existe hijo por confirmar paquete m) THEN
  ACK.SN=m
  Envía_ACK(padre)
  Actualiza_INF_Est(padre)

```
    IF (es un SR) THEN Libera_buffers ENDIF
  ENDIF
  RETURN
```

Suceso: ***Recibe_NAK***

{Recibe desde un hijo asociado al interfaz "i" un NAK_n. Si es un SR pasa al estado de Retransmisión; en caso contrario, pasa al estado de NAK Pendiente}

```
  Actualiza_INF_Est(hijo)
  IF (T_OS activado) THEN PURGE (PILA_paq_fuera_secuencia) ENDIF
  IF (es un SR) THEN
    ESTADO (Retransmisión)
    SUBESTADO (Retransmisión por Solicitud Explícita)_Interfaz[i]_desde_paq[n]
  ELSE              {es un BR}
    Envía_NAK(padre)
    Actualiza_INF_Est(padre)
    ESTADO (NAK Pendiente)
  ENDIF
  RETURN
```

Suceso: ***Recibe_TS***

```
  CASE (flag activo) IN
                                          {Un hijo informa de un cambio de interfaz}
    NI:  Modificación_INF_de_TIB(hijo)
         TS_ACK.NI = Activado
         Envía_TS_ACK(hijo)
                                                    {Se incorpora un miembro}
    NC:  Alta_INF_de_TIB(nuevo_hijo)          {Da de alta en la TIB al nuevo hijo}
         Alta_INF_Est(nuevo_hijo)          {Actualiza su información de estado}
         TS_ACK.NC = Activado
         TS_ACK.RI = Último_RI                      {Informa del último reinicio}
         TS_ACK.RS = Estado_RI          {Informa del estado del último reinicio}
         TS_ACK.LC = Ultimo_confirmado
         Envía_TS_ACK(nuevo_hijo)
                                          {Un hijo solicita salir del árbol RMNP}
    ML:  Baja_INF_de_TIB(hijo)
         Baja_INF_Est(hijo)
         TS_ACK.ML = Activado
         Envía_TS_ACK(hijo)
                                {Si no tiene hijos solicita salir del árbol RMNP}
         IF (número_hijos = 0) THEN
```

```
                TS.ML = Activado
                Envía_TS(padre)                    {Solicita a su padre la desconexión}
                IF (es un SR) THEN Libera_Buffers ENDIF
                FOR t IN t_temporizadores DO       {Detiene los temporizadores}
                    RESET (t)
                END_FOR
                Calcula_T_WF
                Activa_T_WF
                ESTADO (Salida Árbol Solicitada)
            ENDIF
    ESAC
    RETURN
```

Suceso: ***Recibe_TS_ACK***

{Este suceso sólo es posible, si se ha producido un cambio en el interfaz del padre}

```
    IF (flag NI está activado) and (está activado T_WF) THEN RESET(T_WF)
    ELSE Error ENDIF
    RETURN
```

Suceso: ***Recibe_LIB***

{Distribuye el paquete y comprueba si se ha producido una liberación ordenada o un aborto}

```
    FOR i IN n_interfaces DO
        Difunde_paquete_X_interfaz[i]
    END_FOR
    IF (LIB.OR está activo) THEN
        FOR t IN t_temporizadores DO
            IF NOT(t igual T_LIB) THEN RESET[t] ENDIF
        END_FOR
        ESTADO (Liberación Ordenada)
    ELSE_IF (LIB.AB está activo) THEN
            Libera_recursos
            ESTADO (Fin)
        ENDIF
    ENDIF
    RETURN
```

Suceso: ***Recibe_LIB_ACK***

{En este estado la única posibilidad es que el LIB_ACK lleve el flag INF activado y haya sido enviado por la fuente para confirmar que recibió el aviso de que se ha producido una liberación parcial}

IF (flag INF está activado) and (está activado T_{WS}) THEN RESET(T_{WS})
ELSE Error ENDIF
RETURN

Suceso: ***Expira_TRET***

{Expira T_{RET}[n,i] asociado al paquete "n" en el interfaz "i". Este suceso ocurre sólo en los encaminadores SR}

{Detiene todos los temporizadores del interfaz "i"}
FOR T_{RET} IN t_T_{RET}[n,i] DO
Detiene_T_{RET}
END_FOR

{NUM_VENC_T_{RET}[n,i] indica el número de veces que ha vencido T_{RET}[n,i] y se utiliza para detectar posibles particiones}
Incrementa_ NUM_VENC_T_{RET}[n,i]

{Detecta posibles anomalías como una partición en la interred}
FOR h_i IN n_hijos_interfaz[i] DO
IF (NUM_VENC_T_{RET}[n,i] > MAX_RET_BCR_h_i) and (h_i no ha confirmado n) THEN
Baja_INF_de_TIB(h_i)
Baja_INF_Est(h_i)
LIB.AB = Activado {Informa al hijo de la liberación parcial}
Envía_LIB(h_i)
Liberación_parcial_en_curso = true
ENDIF
IF (todo hijo[i] confirmó m) THEN
Actualiza_límite_inferior_W_{INT}[i] {Actualiza límite de la ventana}
ENDIF
END_FOR
IF Liberación_parcial_en_curso THEN
LIB.INF = Activado {Avisa a la fuente de la liberación parcial}
Envía_LIB(fuente)
Calcula_T_{WS}
Activa_T_{WS}
IF (número_hijos = 0) THEN
Libera_Recursos
ESTADO (Avisando Liberación Parcial)

{Comprueba si el resto de los hijos ya le habían confirmado el paquete recibido}
ELSE_IF NOT(existe hijo por confirmar paquete n) THEN
Libera_buffers
ACK.SN=m

```
            Envía_ACK(padre)
            Actualiza_INF_Est(padre)
        ENDIF
    ENDIF
ENDIF
ESTADO (Retransmisión)
SUBESTADO (Retransmisión por Temporizador)_Interfaz[i]
```

Suceso: *Expira_TOS*

{Descarta los paquetes fuera de secuencia almacenados, envía un NAK a su padre solicitando el próximo paquete en secuencia}

```
PURGE (PILA_paq_fuera_secuencia)
Envía_NAK_paq_esperado(padre)
ESTADO (NAK Pendiente)
```

Suceso: *Expira_TLIB*

{Interrumpe el proceso de distribución básica de paquetes y asume que se ha producido una liberación de la conexión, dejando libres los recursos locales asociados a ésta}

```
Libera_recursos
ESTADO (Fin)
```

Suceso: *Expira_TWF*

{Expira el temporizador de espera del TS_ACK del padre. Esta situación sólo puede deberse a que ha habido un cambio de interfaz en el padre}

```
                                {Si es alcanzado el MAX_TWF se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWF)
TS.NI = Activado                                {Flag de nuevo interfaz activado}
Indica_nuevo_interfaz_en_TS
Incrementa_num_retransmisiones_a_padre
Envía_TS(padre)
Calcula_TWF
Activa_TWF                          {Activa el temporizador de espera TS_ACK}
RETURN
```

Suceso: *Expira_TWS*

{Expira el temporizador de espera de confirmación de la fuente asociado al paquete PAQ. Este paquete asociado al temporizador puede ser un TS con el flag RN activado o un LIB con el flag INF activado}

{Si es alcanzado el MAX_TWS se realiza una liberación parcial}

```
CALL_MOD (Comprueba_MAX_TWS)
Incrementa_num_retransmisiones_a_fuente
Envía_PAQ(fuente)          {Reenvía el paquete que estaba asociado al T_WS que expiró}
Calcula_T_WS
Activa_T_WS                {Activa el temporizador de espera TS_ACK}
RETURN
```

Estado NAK Pendiente

{Este estado solamente es válido para los encaminadores BR. Un BR transita a este estado cuando mediante un NAK, ha solicitado a su padre la retransmisión a partir de un determinado paquete. El encaminador continuará en este estado hasta que todos los NAKs sean satisfechos}

```
WHILE (existan NAK pendientes) DO
  IF (existe suceso) THEN
    CASE suceso IN
      Recibe_DAT:          GOSUB(Recibe_DAT)
      Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
      Recibe_MTTUD_ACK:    GOSUB(Recibe_MTTUD_ACK)
      Recibe_ACK:          GOSUB(Recibe_ACK)
      Recibe_NAK:          GOSUB(Recibe_NAK)
      Recibe_TS:           GOSUB(Recibe_TS)
      Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
      Recibe_LIB:          GOSUB(Recibe_LIB)
      Recibe_LIB_ACK:      GOSUB(Recibe_LIB_ACK)
      Expira_T_OS:         GOSUB(Expira_T_OS)
      Expira_T_LIB:        GOSUB(Expira_T_LIB)
      Expira_T_WF:         GOSUB(Expira_T_WF)
      Expira_T_WS:         GOSUB(Expira_T_WS)
    ESAC
  ENDIF
END_WHILE
ESTADO (Distribución Normal)
```

Suceso: ***Recibe_DAT***

{Recibe desde su padre un DAT_n}

```
                {Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (T_LIB , MAX_VAL_TLIB)
                {Comprueba si se ha iniciado un proceso de reinicio}
CALL_MOD (Comprueba_reinicio)
```

```
                {Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la
                fuente un reinicio}
CALL_MOD (Comprueba_cambio_árbol)
                {Comprueba si ha variado el interfaz que utiliza su padre para mandarle da-
                tos. Si es así, avisará a su padre}
CALL_MOD (Comprueba_interfaz)

                {Primeramente comprueba si el paquete recibido esta dentro de la ventana y
                para ello utiliza el procedimiento Dentro_Ventana_R. Posteriormente, clasifica
                el paquete recibido en: solicitado, fuera de secuencia, duplicado sin confirmar,
                duplicado ya confirmado y no transmitido previamente, y según sea el tipo del
                paquete así será el comportamiento. Para las anteriores clasificaciones se ne-
                cesita la siguiente información de estado:
                    - Pendiente: es el menor de los paquetes pendientes de retransmisión; con-
                    secuentemente es el menor de todos los NAK_pendiente del interfaz [i].
                    - Esperado: es el siguiente paquete en secuencia que se estaba esperando
                    cuando llegó un NAK y se pasó al estado de NAK pendiente.
                    - Ultimo_confirmado: último paquete confirmado a su padre}
IF (Dentro_Ventana_R) THEN
   IF (DAT.SN = Pendiente) THEN
         Actualiza_contador_paquete_pendiente
         CALL Distribuye_paquetes
                                                    {Recibe paquete fuera de secuencia}
   ELSE_IF (DAT.SN < Esperado - 1) and (DAT.SN > Pendiente) THEN
               CALL Fuera_de_secuencia
                                                    {Recibe un duplicado sin confirmar}
         ELSE_IF (DAT.SN < Pendiente) and (DAT.SN > Ultimo_confirmado) THEN
                     CALL Duplicado_NO_confirmado
                                                    {Recibe un duplicado ya confirmado}
               ELSE_IF (DAT.SN ≤ Ultimo_confirmado) THEN
                           CALL Duplicado_YA_confirmado
                                                    {Recibe un paquete no transmitido previamente}
                     ELSE                  {DAT.SN > Esperado}
                           Descarta_paquete
                     ENDIF
               ENDIF
         ENDIF
   ENDIF
ELSE {paquete Fuera de la Ventana}
   Error (fuera ventana)
```

ENDIF
RETURN

PROCEDIMIENTOS DEL SUCESO Recibe_DAT

□ *Proceso* Dentro_Ventana_R

{ENS es el Espacio de números de secuencia}
{W_G es el tamaño de la ventana general}

IF (DAT.SN ≥ mod_{ENS}(Ultimo_confirmado + W_G)) and
(DAT.SN ≤ mod_{ENS}(Esperado - W_G)) THEN
Dentro_Ventana_R = FALSE
ELSE
Dentro_Ventana_R = TRUE
ENDIF
END_Proceso

□ *Proceso* Distribuye_paquetes

{Envía el paquete recibido a los hijos, aplicando filtrado de retransmisiones por solicitud, y después comprueba si existen paquetes fuera de secuencia almacenados temporalmente y por tanto pendientes de ser aceptados}

CALL Envía_paquete_hijos

{El siguiente proceso es análogo al que existe en el estado Distribución Normal. Va buscando si el 1° de la pila es el siguiente paquete a retransmitir, y si lo es, lo saca de la pila y lo envía actualizando de nuevo el contador de paquete pendiente. En el caso de que la pila se quede vacía, detiene el temporizador y continúa la retransmisión}

WHILE ((T_{OS} está activo) and (PILA_NAK_fuera_secuencia.NS = Pendiente)) DO
DAT = GET (PILA_NAK_fuera_secuencia)
Actualiza_contador_paquete_pendiente
CALL Envía_paquete_hijos
IF (PILA_NAK_fuera_secuencia está vacía) THEN RESET (T_{OS}) ENDIF
END_WHILE
END_Proceso

□ *Proceso* Envía_paquete_hijos

{Distribuye el paquete a los hijos aplicando filtrado de retransmisiones por solicitud. Consecuentemente, sólo distribuye el paquete por aquellos interfaces por los cuales había sido solicitado}

DAT.LRV = Identificador_propio

```
FOR i IN n_interfaces DO
        DAT.I = i
        IF (existe NAK_pendiente del interfaz[i]) THEN
                Difunde_paquete[Pendiente]_interfaz[i]
                Actualiza  NAK_pendiente del interfaz [i]
                  {Actualiza número de veces que ha transmitido el paquete "n" por el
                  interfaz "i"}
                Incrementa NUM_RET[n,i]
        ENDIF
    END_FOR
END_Proceso
```

□ *Proceso* Fuera_de_secuencia

{Almacena el paquete en la pila de fuera de secuencia, ordenada de menor a mayor, y activa el temporizador de fuera de secuencia, si no está activado}

```
        PUT (PILA_NAK_fuera_secuencia, DAT)
        ORDENA (PILA_NAK_fuera_secuencia)
        IF NOT (T_OS está activo) THEN Activa_T_OS ENDIF
END_Proceso
```

□ *Proceso* Duplicado_NO_confirmado

{Si ha recibido desde su padre un DAT_n distribuye el paquete por todos sus interfaces hijo realizando la función de filtrado de retransmisión por temporizador. Para decidir se debe filtrar o no utiliza la variable NUM_RET[n,i] que almacena el número de retransmisiones del paquete "n" por el interfaz "i". Posteriormente, intenta detectar posibles particiones en la interred, para esto utiliza como variable auxiliar NUM_RET_TEMP[n,i] que almacena el número de retransmisiones del paquete "n" por el interfaz "i" originadas por el vencimiento de un temporizador}

```
DAT.LRV = Identificador_propio
FOR i IN n_interfaces DO
  DAT.I = i
                                {Aplica Filtrado de Retransmisiones por Temporizador}
  IF (NUM_RET[n,i] ≤ MAX_FIL) OR (NUM_RET[n,i] es par)
     THEN                {filtra}
          IF NOT (paquete_confirmado_todos_hijos_ interfaz[i]) THEN
                Difunde_paquete_X_interfaz[i]
                Incrementa_NUM_RET[n,i]
                Incrementa_ NUM_RET_TEMP[n,i]
          ENDIF
     ELSE                {no filtra}
          Difunde_paquete_X_interfaz[i]
```

```
                Incrementa_NUM_RET[n,i]
                Incrementa_ NUM_RET_TEMP[n,i]
        ENDIF
                                {Detecta posibles anomalías como una partición en la interred}
        FOR hi IN n_hijos_interfaz[i] DO
            Calcula_ MAX_RET_BCR_hi
            IF (NUM_RET_TEMP[n,i] > MAX_RET_BCR_hi) and
                                (hi no ha confirmado el paquete recibido) THEN
                Baja_INF_de_TIB(hi)
                Baja_INF_Est(hi)
                LIB.AB = Activado       {Informa al hijo de la liberación parcial}
                Envía_LIB(hi)
                Liberación_parcial_en_curso = true
            ENDIF
        END_FOR
    END_FOR
        IF Liberación_parcial_en_curso THEN
            LIB.INF = Activado          {Avisa a la fuente de la liberación parcial}
            Envía_LIB(fuente)
            Calcula_Tws
            Activa_Tws
            IF (número_hijos = 0 ) THEN
                ESTADO (Avisando Liberación Parcial)
                        {Comprueba si el resto de los hijos ya le habían confirmado el
                        paquete recibido}
              ELSE_IF NOT(existe hijo por confirmar paquete n) THEN
                ACK.SN=n
                Envía_ACK(padre)
                Actualiza_INF_Est(padre)
              ENDIF
            ENDIF
        ENDIF
    ENDIF
END_Proceso
```

□ *Proceso* Duplicado_YA_confirmado

```
    Descarta_paquete
    ACK.SN = Ultimo_confirmado
    Envía_ACK(padre)
END_Proceso
```

Suceso: ***Recibe_MTTUD***

{Lo envía a sus hijos a través del árbol de distribución}

{Actualiza el temporizador de liberación por silencio de la fuente}

Actualiza (T_{LIB} , MAX_VAL_TLIB)

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}

CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que le envía el padre los paquetes de datos. Si es así avisará a su padre}

CALL_MOD (Comprueba_interfaz)

{Distribuye el paquete por todos los interfaces}

```
MTTUD.LRV = Identificador_propio
FOR i IN n_interfaces DO
   MTTU.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
RETURN
```

Suceso: ***Recibe_MTTUD_ACK***

{Realiza el proceso de agregación de MTTUD}

```
Actualiza_INF_Est(hijo)
IF NOT(existe hijo por confirmar MTTUD) THEN
   Envía_MTTUD_ACK          {Envía confirmación del paquete MTTUD al padre}
ENDIF
RETURN
```

Suceso: ***Recibe_ACK***

{Recibe desde un hijo asociado al interfaz "i" un ACK_n. Actualiza la información de estado asociada al hijo que le envió el asentimiento; y si procede realiza un proceso de agregación de ACKs}

Actualiza_INF_Est(hijo)

{Proceso de agregación de ACKs. Comprueba el paquete mínimo "m" ≤ "n" confirmado por todos los hijos, y envía un ACK al padre}

```
IF NOT(existe hijo por confirmar paquete m) THEN
   ACK.SN=m
   Envía_ACK(padre)
   Actualiza_INF_Est(padre)
ENDIF
RETURN
```

Suceso: ***Recibe_NAK***

{Recibe desde un hijo asociado al interfaz "i" un NAK_n}

{Actualiza la información de estado asociada al hijo del interfaz "i", si éste no tenía ninguna solicitud de retransmisión pendiente, o si la tenía, era a partir de un paquete mayor que "n"}

```
IF (NOT(existe NAK_pendiente del interfaz[i]) OR (NAK_pendiente del interfaz[i]> n))
THEN
   Actualiza NAK_pendiente del interfaz[i]
ENDIF
IF (Último_solicitado_padre > NAK_pendiente del interfaz[i]) THEN
   Envía_NAK_n(padre)                                    {Filtrado de NAKs}
ENDIF
RETURN
```

Suceso: ***Recibe_TS***

```
CASE (flag activo) IN
                                          {Un hijo informa de un cambio de interfaz}
   NI:  Modificación_INF_de_TIB(hijo)
        TS_ACK.NI = Activado
        Envía_TS_ACK(hijo)
                                                    {Se incorpora un miembro}
   NC:  Alta_INF_de_TIB(nuevo_hijo)          {Da de alta en la TIB al nuevo hijo}
        Alta_INF_Est(nuevo_hijo)             {Actualiza su información de estado}
        TS_ACK.NC = Activado
        TS_ACK.RI = Último_RI                          {Informa del último reinicio}
        TS_ACK.RS = Estado_RI                {Informa del estado del último reinicio}
        TS_ACK.LC = Ultimo_confirmado
        Envía_TS_ACK(nuevo_hijo)
                                          {Un hijo solicita salir del árbol RMNP}
   ML:  Baja_INF_de_TIB(hijo)
        Baja_INF_Est(hijo)
        TS_ACK.ML = Activado
        Envía_TS_ACK(hijo)
                                   {Si no tiene hijos solicita salir del árbol RMNP}
        IF (número_hijos = 0) THEN
              TS.ML = Activado
              Envía_TS(padre)              {Solicita a su padre la desconexión}
              FOR t IN t_temporizadores DO          {Detiene los temporizadores}
                    RESET (t)
              END_FOR
```

Calcula_T_{WF}
Activa_T_{WF}
ESTADO (Salida Árbol Solicitada)
ENDIF
ESAC
RETURN

Suceso: ***Recibe_TS_ACK***

{Este suceso sólo es posible, si se ha producido un cambio en el interfaz del padre}

IF (flag NI está activado) and (está activado T_{WF}) THEN RESET(T_{WF})
ELSE Error ENDIF
RETURN

Suceso: ***Recibe_LIB***

{Distribuye el paquete y comprueba si se ha producido una liberación ordenada o un aborto}

FOR i IN n_interfaces DO
Difunde_paquete_X_interfaz[i]
END_FOR
IF (LIB.OR está activo) THEN
FOR t IN t_temporizadores DO
IF NOT(t igual T_{LIB}) THEN RESET[t] ENDIF
END_FOR
ESTADO (Liberación Ordenada)
ELSE_IF (LIB.AB está activo) THEN
Libera_recursos
ESTADO (Fin)
ENDIF
ENDIF
RETURN

Suceso: ***Recibe_LIB_ACK***

{Este estado la única posibilidad es que este LIB_ACK lleve el flag INF activado y haya sido enviado por la fuente para confirmar que recibió el aviso de que se ha producido una liberación parcial}

IF (flag INF está activado) and (está activado T_{WS}) THEN RESET(T_{WS})
ELSE Error ENDIF
RETURN

Suceso: ***Expira_TOS***

{Descarta los paquetes fuera de secuencia almacenados y envía un NAK a su padre solicitando el próximo paquete en secuencia }

PURGE (PILA_NAK_fuera_secuencia)
Envía_NAK_paq_esperado(padre)
RETURN

Suceso: ***Expira_TLIB***

{Interrumpe el proceso de distribución básica de paquetes, y el encaminador asume que se ha producido una liberación de la conexión y deja libres los recursos locales asociados a ésta}

Libera_recursos
ESTADO (Fin)

Suceso: ***Expira_TWF***

{Expira el temporizador de espera del TS_ACK del padre. Esta situación sólo puede deberse a que ha habido un cambio de interfaz en el padre}

{Si es alcanzado el MAX_TWF se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWF)
TS.NI = Activado {Flag de nuevo interfaz activado}
Indica_nuevo_interfaz_en_TS
Incrementa_num_retransmisiones_a_padre
Envía_TS(padre)
Calcula_T_{WF}
Activa_T_{WF} {Activa el temporizador de espera TS_ACK}
RETURN

Suceso: ***Expira_TWS***

{Expira el temporizador de espera de confirmación de la fuente asociado al paquete PAQ. Este paquete asociado al temporizador puede ser un TS con el flag RN activado o un LIB con el flag INF activado}

{Si es alcanzado el MAX_TWS se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWS)
Incrementa_num_retransmisiones_a_fuente
Envía_PAQ(fuente) {Reenvía el paquete que estaba asociado al T_{WS} que expiró}
Calcula_T_{WS}
Activa_T_{WS} {Activa el temporizador de espera TS_ACK}
RETURN

Estado Retransmisión

{Este estado solamente es válido para los SRs. Un SR transita a este estado cuando en alguno de sus interfaces hijo se inicia un proceso de retransmisión ya sea por solicitud o por temporizador. El encaminador continuará en este estado mientras que al menos en uno de sus interfaces hijo esté en marcha un proceso de retransmisión. Los interfaces con un proceso de retransmisión activo estarán o en el subestado de retransmisión por temporizador o en el subestado de retransmisión por solicitud explícita. Por el contrario los interfaces que no tienen asociado ningún proceso de retransmisión no tendrán asociado ningún subestado especial y estarán en el estado retransmisión}

```
WHILE (exista proceso_retransmisión[i]) DO
  IF (existe suceso) THEN
    CASE suceso IN
      Recibe_DAT:          GOSUB(Recibe_DAT)
      Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
      Recibe_MTTUD_ACK:    GOSUB(Recibe_MTTUD_ACK)
      Recibe_ACK:          GOSUB(Recibe_ACK)
      Recibe_NAK:          GOSUB(Recibe_NAK)
      Recibe_TS:           GOSUB(Recibe_TS)
      Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
      Recibe_LIB:          GOSUB(Recibe_LIB)
      Recibe_LIB_ACK:      GOSUB(Recibe_LIB_ACK)
      Expira_T_RET:        GOSUB(Expira_T_RET)
      Expira_T_LIB:        GOSUB(Expira_T_LIB)
      Expira_T_WF:         GOSUB(Expira_T_WF)
      Expira_T_WS:         GOSUB(Expira_T_WS)
    ESAC
  ENDIF
END_WHILE
ESTADO (Distribución Normal)
```

Suceso: *Recibe_DAT*

{Descarta el paquete, y si es un duplicado previamente confirmado reenvía al padre RMNP el último ACK}

Actualiza (T_{LIB} , MAX_VAL_TLIB)

{Comprueba si se ha iniciado un proceso de reinicio}

CALL_MOD (Comprueba_reinicio)

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}

CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que le envía el padre los paquetes de datos. Si es así avisará a su padre}

CALL_MOD (Comprueba_interfaz)
Descarta_paquete
IF (DAT es duplicado_confirmado) THEN Envía_ACK_último ENDIF
RETURN

Suceso: ***Recibe_MTTUD***

{Lo envía a sus hijos a través del árbol de distribución}

{Actualiza el temporizador de liberación por silencio de la fuente}

Actualiza (T_{LIB} , MAX_VAL_TLIB)

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}

CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que le envía el padre los paquetes de datos. Si es así avisará a su padre}

CALL_MOD (Comprueba_interfaz)

{Distribuye el paquete por todos los interfaces}

MTTUD.LRV = Identificador_propio
FOR i IN n_interfaces DO
  MTTU.I = i
  Difunde_paquete_X_interfaz[i]
END_FOR
RETURN

Suceso: ***Recibe_MTTUD_ACK***

{Realiza el proceso de agregación de MTTUD}

Actualiza_INF_Est(hijo)
IF NOT(existe hijo por confirmar MTTUD) THEN
  Envía_MTTUD_ACK {Envía confirmación del paquete MTTUD al padre}
ENDIF
RETURN

Suceso: ***Recibe_ACK***

{Recibe desde un hijo asociado al interfaz "i" un ACK_n. Actualiza la información de estado asociada al hijo que le envió el asentimiento, y si procede realiza un proceso de agregación de ACKs}

Actualiza_INF_Est(hijo)

{Si todos los hijos accesibles por el interfaz "i" han confirmado el paquete "m" (siendo "m" ≤ "n"), detiene el temporizador de retransmisión asociado "i" y actualiza el límite de la ventana}

IF (todo hijo[i] confirmó m) and (T_{RET}[m,i] está activo) THEN
  RESET (T_{RET}[m,i])
  Actualiza_límite_inferior_W_{INT}[i] {Actualiza límite de la ventana}
ENDIF

{Proceso de agregación de ACKs. Comprueba el paquete mínimo "m" ≤ "n" confirmado por todos los hijos, y envía un ACK al padre}

IF NOT(existe hijo por confirmar paquete m) THEN
  ACK.SN=m
  Envía_ACK(padre)
  Actualiza_INF_Est(padre)
  Libera_buffers
ENDIF
RETURN

Suceso: ***Recibe_NAK***

{Recibe desde un hijo asociado al interfaz "i" un NAK_n, dicho interfaz pasa el subestado de retransmisión por solicitud explícita}

Actualiza_INF_Est(hijo)
SUBESTADO (Retransmisión por Solicitud Explícita)_Interfaz[i]_desde_paq[n]

Suceso: ***Recibe_TS***

CASE (flag activo) IN

{Un hijo informa de un cambio de interfaz}

NI: Modificación_INF_de_TIB(hijo)
  TS_ACK.NI = Activado
  Envía_TS_ACK(hijo)

{Se incorpora un miembro}

NC: Alta_INF_de_TIB(nuevo_hijo) {Da de alta en la TIB al nuevo hijo}
  Alta_INF_Est(nuevo_hijo) {Actualiza su información de estado}
  TS_ACK.NC = Activado
  TS_ACK.RI = Último_RI {Informa del último reinicio}
  TS_ACK.RS = Estado_RI {Informa del estado del último reinicio}
  TS_ACK.LC = Ultimo_confirmado
  Envía_TS_ACK(nuevo_hijo)

{Un hijo solicita salir del árbol RMNP}

ML: Baja_INF_de_TIB(hijo)
  Baja_INF_Est(hijo)

```
            TS_ACK.ML = Activado
            Envía_TS_ACK(hijo)
                                        {Si no tiene hijos solicita salir del árbol RMNP}
            IF (número_hijos = 0) THEN
                  TS.ML = Activado
                  Envía_TS(padre)              {Solicita a su padre la desconexión}
                  Libera_Buffers
                  FOR t IN t_temporizadores DO        {Detiene los temporizadores}
                        RESET (t)
                  END_FOR
                  Calcula_T_WF
                  Activa_T_WF
                  ESTADO (Salida Árbol Solicitada)
            ENDIF
      ESAC
      RETURN
```

Suceso: ***Recibe_TS_ACK***

{Este suceso sólo es posible, si se ha producido un cambio en el interfaz del padre}

```
      IF (flag NI está activado) and (está activado T_WF) THEN RESET(T_WF)
      ELSE Error ENDIF
      RETURN
```

Suceso: ***Recibe_LIB***

{Distribuye el paquete y comprueba si se ha producido una liberación ordenada o un aborto}

```
      FOR i IN n_interfaces DO
         Difunde_paquete_X_interfaz[i]
      END_FOR
      IF (LIB.OR está activo) THEN
         FOR t IN t_temporizadores DO
               IF NOT(t igual T_LIB) THEN RESET[t] ENDIF
         END_FOR
         ESTADO (Liberación Ordenada)
      ELSE_IF (LIB.AB está activo) THEN
               Libera_recursos
               ESTADO (Fin)
          ENDIF
      ENDIF
      RETURN
```

Suceso: ***Recibe_LIB_ACK***

{En este estado la única posibilidad es que el LIB_ACK lleve el flag INF activado y haya sido enviado por la fuente para confirmar que recibió el aviso de que se ha producido una liberación parcial}

IF (flag INF está activado) and (está activado T_{WS}) THEN RESET(T_{WS})
ELSE Error ENDIF
RETURN

Suceso: ***Expira_TRET***

{Expira T_{RET}[n,i] asociado al paquete "n" en el interfaz "i"}

{Detiene todos los temporizadores del interfaz "i"}
FOR T_{RET} IN t_T_{RET}[n,i] DO
  Detiene_T_{RET}
END_FOR

{NUM_VENC_T_{RET}[n,i] indica el número de veces que ha vencido T_{RET}[n,i] y se utiliza para detectar posibles particiones}
Incrementa_ NUM_VENC_T_{RET}[n,i]

{Detecta posibles anomalías como una partición en la interred}
FOR h_i IN n_hijos_interfaz[i] DO
  IF (NUM_VENC_T_{RET}[n,i] > MAX_RET_BCR_h_i) and (h_i no ha confirmado n) THEN
    Baja_INF_de_TIB(h_i)
    Baja_INF_Est(h_i)
    LIB.AB = Activado {Informa al hijo de la liberación parcial}
    Envía_LIB(h_i)
    Liberación_parcial_en_curso = true
  ENDIF
  IF (todo hijo[i] confirmó m) THEN
    Actualiza_límite_inferior_W_{INT}[i] {Actualiza límite de la ventana}
  ENDIF
END_FOR
IF Liberación_parcial_en_curso THEN
  LIB.INF = Activado {Avisa a la fuente de la liberación parcial}
  Envía_LIB(fuente)
  Calcula_T_{WS}
  Activa_T_{WS}
  IF (número_hijos = 0) THEN
    Libera_Recursos
    ESTADO (Avisando Liberación Parcial)

{Comprueba si el resto de los hijos ya le habían confirmado el paquete recibido}
ELSE_IF NOT(existe hijo por confirmar paquete n) THEN
Libera_buffers
ACK.SN=m
Envía_ACK(padre)
Actualiza_INF_Est(padre)
ENDIF
ENDIF
ENDIF
SUBESTADO (Retransmisión por Temporizador)_Interfaz[i]

Suceso: *Expira_T_{LIB}*

{Interrumpe el proceso de distribución básica de paquetes y asume que se ha producido una liberación de la conexión, dejando libres los recursos locales asociados a ésta}

Libera_recursos
ESTADO (Fin)

Suceso: *Expira_T_{WF}*

{Expira el temporizador de espera del TS_ACK del padre. Esta situación sólo puede deberse a que ha habido un cambio de interfaz en el padre}

{Si es alcanzado el MAX_TWF se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWF)
TS.NI = Activado {Flag de nuevo interfaz activado}
Indica_nuevo_interfaz_en_TS
Incrementa_num_retransmisiones_a_padre
Envía_TS(padre)
Calcula_T_{WF}
Activa_T_{WF} {Activa el temporizador de espera TS_ACK}
RETURN

Suceso: *Expira_T_{WS}*

{Expira el temporizador de espera de confirmación de la fuente asociado al paquete PAQ. Este paquete asociado al temporizador puede ser un TS con el flag RN activado o un LIB con el flag INF activado}

{Si es alcanzado el MAX_TWS se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWS)
Incrementa_num_retransmisiones_a_fuente
Envía_PAQ(fuente) {Reenvía el paquete que estaba asociado al T_{WS} que expiró}

```
Calcula_Tws
Activa_Tws                          {Activa el temporizador de espera TS_ACK}
RETURN
```

Subestado Retransmisión por Temporizador

{Tras el vencimiento de un temporizador de retransmisión asociado al interfaz "i", dicho interfaz pasa al subestado de retransmisión por temporizador y comienza retransmitir todos los paquetes pendientes de confirmación, siendo próximo_a_enviar el primero de éstos. Con el fin de detectar que algún hijo se haya cambiado de interfaz, en el caso de que el primer paquete pendiente de retransmitir ya haya sido transmitido por el interfaz "i" más de MAX_FIL veces, irán alternándose procesos de retransmisión en los que únicamente se retransmite por el interfaz "i" y procesos de retransmisión en los que se retransmite por todos los interfaces hijo}

```
DAT.LRV = Identificador_propio
            {Comprueba si debe enviar los paquetes retransmitidos únicamente por el interfaz "i" o por todos los interfaces}
IF (NUM_RET[próximo_a_enviar,i] ≤ MAX_FIL) or
                                (NUM_RET[próximo_a_enviar,i] es par) THEN
   FOR próximo_a_enviar = 1°_paq_no_confirmado TO último_paq_enviado DO
      DAT.I = i
      Difunde_paquete_[próximo_a_enviar]_X_interfaz[i]
      Incrementa_NUM_RET[n,i]
      Calcula_T_RET[próximo_a_enviar,i]
      Activa_T_RET[próximo_a_enviar,i]
      IF NUM_RET[próximo_a_enviar,i] > MAX_RET_BWD THEN
            Proceso_Ajuste_W_INT[i]
      ENDIF

      IF (existe suceso) THEN
         CASE suceso IN
            Recibe_MTTUD_ACK:     GOSUB(Recibe_MTTUD_ACK)
            Recibe_ACK:           GOSUB(Recibe_ACK)
            Recibe_NAK:           GOSUB(Recibe_NAK)
            Recibe_TS:            GOSUB(Recibe_TS)
            Recibe_LIB_ACK:       GOSUB(Recibe_LIB_ACK)
         ESAC
      ENDIF
ELSE                              {Lo retransmite por todos los interfaces}
   FOR j IN n_interfaces DO
      FOR próximo_a_enviar = 1°_paq_no_confirmado TO último_paq_enviado DO
```

```
                DAT.I = j
                Difunde_paquete_[próximo_a_enviar]_X_interfaz[j]
                Incrementa_NUM_RET[próximo_a_enviar,j]
                Calcula_T_RET[próximo_a_enviar,j]
                Activa_T_RET[próximo_a_enviar,j]
                IF NUM_RET[próximo_a_enviar,j] > MAX_RET_BWD THEN
                        Proceso_Ajuste_W_INT[j]
                ENDIF

                IF (existe suceso) THEN
                   CASE suceso IN
                        Recibe_MTTUD_ACK:       GOSUB(Recibe_MTTUD_ACK)
                        Recibe_ACK:             GOSUB(Recibe_ACK)
                        Recibe_NAK:             GOSUB(Recibe_NAK)
                        Recibe_TS:              GOSUB(Recibe_TS)
                        Recibe_LIB_ACK:         GOSUB(Recibe_LIB_ACK)
                   ESAC
                ENDIF
            END_FOR
        END_FOR
    ENDIF
    FIN_SUBESTADO (Retransmisión por Temporizador)_Interfaz[i]
```

Suceso: ***Recibe_MTTUD_ACK***

{Realiza el proceso de agregación de MTTUD}

```
Actualiza_INF_Est(hijo)
IF NOT(existe hijo por confirmar MTTUD) THEN
   Envía_MTTUD_ACK          {Envía confirmación del paquete MTTUD al padre}
ENDIF
RETURN
```

Suceso: ***Recibe_ACK***

{Recibe desde un hijo asociado al interfaz "i" un ACK_n. Actualiza la información de estado asociada al hijo que le envió el asentimiento, y si procede realiza un proceso de agregación de ACKs, posteriormente reanuda el proceso de retransmisión a partir del primer paquete no confirmado}

```
Actualiza_INF_Est(hijo)
        {Si todos los hijos accesibles por el interfaz "i" han confirmado el paquete
        "m" (siendo "m" ≤ "n"), detiene el temporizador de retransmisión asociado "i"
        y actualiza el límite de la ventana}
```

IF (todo hijo[i] confirmó m) and (T_{RET}[m,i] está activo) THEN
  RESET (T_{RET}[m,i])
  Actualiza_límite_inferior_W_{INT}[i] {Actualiza límite de la ventana}
ENDIF

{Proceso de agregación de ACKs. Comprueba el paquete mínimo "m" ≤ "n" confirmado por todos los hijos, y envía un ACK al padre}

IF NOT(existe hijo por confirmar paquete m) THEN
  ACK.SN=m
  Envía_ACK(padre)
  Actualiza_INF_Est(padre)
  Libera_buffers
ENDIF

{Si todos los hijos han confirmado hasta el paquete "m" continuará la retransmisión a partir de éste}

IF (próximo_a_enviar < m) THEN
  próximo_a_enviar = m + 1
ENDIF
RETURN

Suceso: ***Recibe_NAK***

{Descarta el paquete y sigue en este estado}

Descarta_paquete
RETURN

Suceso: ***Recibe_TS***

CASE (flag activo) IN

{Un hijo informa de un cambio de interfaz}

NI: Modificación_INF_de_TIB(hijo)
TS_ACK.NI = Activado
Envía_TS_ACK(hijo)

{Se incorpora un miembro}

NC: Alta_INF_de_TIB(nuevo_hijo) {Da de alta en la TIB al nuevo hijo}
Alta_INF_Est(nuevo_hijo) {Actualiza su información de estado}
TS_ACK.NC = Activado
TS_ACK.RI = Último_RI {Informa del último reinicio}
TS_ACK.RS = Estado_RI {Informa del estado del último reinicio}
TS_ACK.LC = Ultimo_confirmado
Envía_TS_ACK(nuevo_hijo)

{Un hijo solicita salir del árbol RMNP}

ML: Baja_INF_de_TIB(hijo)

```
            Baja_INF_Est(hijo)
            TS_ACK.ML = Activado
            Envía_TS_ACK(hijo)
                                        {Si no tiene hijos solicita salir del árbol RMNP}
            IF (número_hijos = 0) THEN
                  TS.ML = Activado
                  Envía_TS(padre)              {Solicita a su padre la desconexión}
                  Libera_Buffers
                  FOR t IN t_temporizadores DO          {Detiene los temporizadores}
                        RESET (t)
                  END_FOR
                  Calcula_T_WF
                  Activa_T_WF
                  ESTADO (Salida Árbol Solicitada)
            ENDIF
      ESAC
      RETURN
```

Suceso: ***Recibe_LIB_ACK***

{En este estado la única posibilidad es que el LIB_ACK lleve el flag INF activado y haya sido enviado por la fuente para confirmar que recibió el aviso de que se ha producido una liberación parcial}

```
IF (flag INF está activado) and (está activado T_WS) THEN RESET(T_WS)
ELSE Error ENDIF
RETURN
```

Subestado Retransmisión por Solicitud Explícita

{Tras recibir un NAK_n desde un hijo asociado al interfaz "i", dicho interfaz pasa al subestado de retransmisión por solicitud explícita y comienza retransmitir todos los paquetes pendientes de confirmación a parteir del paquete "n"}

```
DAT.LRV = Identificador_propio
FOR próximo_a_enviar = n TO último_paq_enviado DO
   DAT.I = i
   Difunde_paquete[próximo_a_enviar]_interfaz[i]
   Incrementa_NUM_RET[n,i]
   Calcula_T_RET[próximo_a_enviar,i]
   Activa_T_RET[próximo_a_enviar,i]
   Actualiza NAK_pendiente del interfaz [i]
   IF NUM_RET[próximo_a_enviar,i] > MAX_RET_BWD THEN
```

```
        Proceso_Ajuste_W_INT[i]
    ENDIF

    IF (existe suceso) THEN
        CASE suceso IN
            Recibe_DAT:              GOSUB(Recibe_DAT)
            Recibe_MTTUD_ACK:        GOSUB(Recibe_MTTUD_ACK)
            Recibe_ACK:              GOSUB(Recibe_ACK)
            Recibe_NAK:              GOSUB(Recibe_NAK)
            Recibe_TS:               GOSUB(Recibe_TS)
            Recibe_LIB_ACK:          GOSUB(Recibe_LIB_ACK)
            Expira_T_RET:            GOSUB(Expira_T_RET)
        ESAC
    ENDIF
END_FOR
FIN_SUBESTADO (Retransmisión por Solicitud Explícita)_Interfaz[i]
```

Suceso: ***Recibe_DAT***

{Descarta el paquete, y si es un duplicado previamente confirmado reenvía al padre RMNP el último ACK}

```
Actualiza (T_LIB , MAX_VAL_TLIB)
                                {Comprueba si se ha iniciado un proceso de reinicio}
CALL_MOD (Comprueba_reinicio)
            {Comprueba si ha variado el padre que le envía datos}
CALL_MOD (Comprueba_cambio_árbol)
            {Comprueba si ha variado el interfaz por el que le envía el padre los paquetes
            de datos si es así avisará a su padre}
CALL_MOD (Comprueba_interfaz)
Descarta_paquete
IF (DAT es duplicado_confirmado) THEN Envía_ACK_último ENDIF
RETURN
```

Suceso: ***Recibe_MTTUD_ACK***

{Realiza el proceso de agregación de MTTUD}

```
Actualiza_INF_Est(hijo)
IF NOT(existe hijo por confirmar MTTUD) THEN
    Envía_MTTUD_ACK          {Envía confirmación del paquete MTTUD al padre}
ENDIF
RETURN
```

Suceso: ***Recibe_ACK***

{Recibe desde un hijo asociado al interfaz "i" un ACK_n. Actualiza la información de estado asociada al hijo que le envió el asentimiento, y si procede realiza un proceso de agregación de ACKs, posteriormente reanuda el proceso de retransmisión a partir del primer paquete no confirmado}

```
Actualiza_INF_Est(hijo)
        {Si todos los hijos accesibles por el interfaz "i" han confirmado el paquete
        "m" (siendo "m" ≤ "n"), detiene el temporizador de retransmisión asociado "i"
        y actualiza el límite de la ventana}
IF (todo hijo[i] confirmó m) and (T_RET[m,i] está activo) THEN
   RESET (T_RET[m,i])
   Actualiza_límite_inferior_W_INT[i]          {Actualiza límite de la ventana}
ENDIF
        {Proceso de agregación de ACKs. Comprueba el paquete mínimo "m" ≤ "n"
        confirmado por todos los hijos, y envía un ACK al padre}
IF NOT(existe hijo por confirmar paquete m) THEN
   ACK.SN=m
   Envía_ACK(padre)
   Actualiza_INF_Est(padre)
   Libera_buffers
ENDIF
        {Si todos los hijos han confirmado hasta el paquete "m" continuará la re-
        transmisión a partir de éste}
IF (próximo_a_enviar < m) THEN
   próximo_a_enviar = m + 1
ENDIF
RETURN
```

Suceso: ***Recibe_NAK***

{Recibe desde un hijo asociado al interfaz "i" un NAK_n}

```
        {Actualiza la información de estado asociada al hijo del interfaz "i", si éste no
        tenía ninguna solicitud de retransmisión pendiente, o si la tenía, era a partir de
        un paquete mayor que "n"}
IF (NOT(existe NAK_pendiente del interfaz[i]) OR (NAK_pendiente del interfaz[i]> n))
THEN
   Actualiza NAK_pendiente del interfaz[i]
ENDIF
        {Si el NAK recibido solicita un paquete anterior al próximo a enviar se co-
        mienza un nuevo proceso de retransmisión a partir del "n"}
```

```
IF (n < próximo_a_enviar) THEN
   SUBESTADO (Retransmisión por Solicitud Explícita)_Interfaz[i]_desde_paq[n]
ELSE
   RETURN
ENDIF
```

Suceso: ***Recibe_TS***

```
CASE (flag activo) IN
                                                {Un hijo informa de un cambio de interfaz}
   NI:  Modificación_INF_de_TIB(hijo)
        TS_ACK.NI = Activado
        Envía_TS_ACK(hijo)
                                                {Se incorpora un miembro}
   NC:  Alta_INF_de_TIB(nuevo_hijo)             {Da de alta en la TIB al nuevo hijo}
        Alta_INF_Est(nuevo_hijo)                {Actualiza su información de estado}
        TS_ACK.NC = Activado
        TS_ACK.RI = Último_RI                   {Informa del último reinicio}
        TS_ACK.RS = Estado_RI                   {Informa del estado del último reinicio}
        TS_ACK.LC = Ultimo_confirmado
        Envía_TS_ACK(nuevo_hijo)
                                                {Un hijo solicita salir del árbol RMNP}
   ML:  Baja_INF_de_TIB(hijo)
        Baja_INF_Est(hijo)
        TS_ACK.ML = Activado
        Envía_TS_ACK(hijo)
                                                {Si no tiene hijos solicita salir del árbol RMNP}
        IF (número_hijos = 0) THEN
              TS.ML = Activado
              Envía_TS(padre)                   {Solicita a su padre la desconexión}
              Libera_Buffers
              FOR t IN t_temporizadores DO      {Detiene los temporizadores}
                    RESET (t)
              END_FOR
              Calcula_T_WF
              Activa_T_WF
              ESTADO (Salida Árbol Solicitada)
        ENDIF
ESAC
RETURN
```

Suceso: ***Recibe_LIB_ACK***

{En este estado la única posibilidad es que el LIB_ACK lleve el flag INF activado y haya sido enviado por la fuente para confirmar que recibió el aviso de que se ha producido una liberación parcial}

IF (flag INF está activado) and (está activado T_{WS}) THEN RESET(T_{WS})
ELSE Error ENDIF
RETURN

Suceso: ***Expira_TRET***

{Expira T_{RET}[n,i] asociado al paquete "n" en el interfaz "i"}

{Detiene todos los temporizadores del interfaz "i"}
FOR T_{RET} IN t_T_{RET}[n,i] DO
  Detiene_T_{RET}
END_FOR

{NUM_VENC_T_{RET}[n,i] indica el número de veces que ha vencido T_{RET}[n,i] y se utiliza para detectar posibles particiones}
Incrementa_ NUM_VENC_T_{RET}[n,i]

{Detecta posibles anomalías como una partición en la interred}
FOR h_i IN n_hijos_interfaz[i] DO
  IF (NUM_VENC_T_{RET}[n,i] > MAX_RET_BCR_h_i) and (h_i no ha confirmado n) THEN
    Baja_INF_de_TIB(h_i)
    Baja_INF_Est(h_i)
    LIB.AB = Activado {Informa al hijo de la liberación parcial}
    Envía_LIB(h_i)
    Liberación_parcial_en_curso = true
  ENDIF
  IF (todo hijo[i] confirmó m) THEN
    Actualiza_límite_inferior_W_{INT}[i] {Actualiza límite de la ventana}
  ENDIF
END_FOR
IF Liberación_parcial_en_curso THEN
  LIB.INF = Activado {Avisa a la fuente de la liberación parcial}
  Envía_LIB(fuente)
  Calcula_T_{WS}
  Activa_T_{WS}
  IF (número_hijos = 0) THEN
    Libera_Recursos
    ESTADO (Avisando Liberación Parcial)

{Comprueba si el resto de los hijos ya le habían confirmado el paquete recibido}

```
      ELSE_IF NOT(existe hijo por confirmar paquete n) THEN
          Libera_buffers
          ACK.SN=m
          Envía_ACK(padre)
          Actualiza_INF_Est(padre)
      ENDIF
    ENDIF
ENDIF
```

{Si ha vencido el temporizador asociado a un paquete anterior al primero que ha sido retransmitido se inicia una retransmisión por temporizador}

```
IF (Primero_enviado > n) THEN
   SUBESTADO (Retransmisión por Temporizador)_Interfaz[i]
ENDIF
RETURN
```

Estado Inicio

{Inicializa variables, prepara buffers y memoria de almacenamiento para la información de estado, y crea la TIB para su posterior uso desde el establecimiento de la conexión hasta su liberación}

```
BUCLE_INFINITO
   WHILE NOT(existe suceso) DO
        CALL Proceso_ocioso
   END_WHILE
     CASE suceso IN
        Recibe_DAT:          GOSUB(Recibe_DAT)
        Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
     ESAC
FIN_BUCLE
```

Suceso: ***Recibe_DAT***

{Como es el primer paquete enviado por la fuente, lo envía a sus hijos a través del árbol de distribución e inicia el proceso de unión al árbol RMNP}

{Actualiza el temporizador de liberación por silencio de la fuente}

```
Actualiza (T_LIB , MAX_VAL_TLIB)
Alta_INF_de_TIB(Padre)
Alta_INF_de_Est(Padre)
```

{Distribuye el paquete por todos los interfaces}

```
DAT.LRV = Identificador_propio
```

```
FOR i IN n_interfaces DO
   DAT.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
                                        {Solicita incorporarse al árbol RMNP}
TS.NC = Activado
Envía_TS(padre)
Calcula_T_WF
Activa_T_WF
ESTADO (Pendiente Entrar Árbol)
```

Suceso: ***Recibe_MTTUD***

{Como es el primer paquete enviado por la fuente, lo envía a sus hijos a través del árbol de distribución e inicia el proceso de unión al árbol RMNP}

```
                        {Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (T_LIB , MAX_VAL_TLIB)
Alta_INF_de_TIB(Padre)
Alta_INF_de_Est(Padre)
                                        {Distribuye el paquete por todos los interfaces}
MTTUD.LRV = Identificador_propio
FOR i IN n_interfaces DO
   MTTU.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
                                        {Solicita a su padre entrar en el árbol}
TS.NC = Activado
Envía_TS(padre)
Calcula_T_WF
Activa_T_WF
ESTADO (Pendiente Entrar Árbol)
```

Estado Pendiente Entrar Árbol

{En este estado el encaminador se une al árbol de distribución RMNP, a la vez que va recogiendo las peticiones de otros encaminadores o miembros que se encuentran localizables a través de sus interfaces hijos. El proceso inicial de construcción del árbol termina, cuando todos los miembros reciben el TS_ACK correspondiente. En este estado decide si va a ser un SR o un BR y reserva recursos e inicializa las pilas necesarias dependiendo de si es uno u otro}

```
BUCLE_INFINITO
```

```
    WHILE NOT(existe suceso) DO
        CALL Proceso_ocioso
    END_WHILE
      CASE suceso IN
        Recibe_DAT:          GOSUB(Recibe_DAT)
        Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
        Recibe_TS:           GOSUB(Recibe_TS)
        Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
        Recibe_LIB:          GOSUB(Recibe_LIB)
        Expira_T_LIB:        GOSUB(Expira_T_LIB)
        Expira_T_WF:         GOSUB(Expira_T_WF)
      ESAC
FIN_BUCLE
```

Suceso: ***Recibe_DAT***

{Distribuye el paquete por todos los interfaces e intenta detectar el final de la fase de establecimiento, lo cual significa que buscará dos paquetes distintos con el flag CR desactivado}

```
                                    {Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (T_LIB , MAX_VAL_TLIB)
                                    {Distribuye el paquete por todos los interfaces}
DAT.LRV = Identificador_propio
FOR i IN n_interfaces DO
   DAT.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
                                    {Comprueba si es el primero con el CR desactivado}
IF (DAT.CR = Desactivado) and NOT(1er_paquete_CR_desactivado) THEN
        1er_paquete_CR_desactivado = true
            {Comprueba si es un segundo paquete recibido con CR desactivado, y si ambos paquetes son distintos}
ELSE_IF (DAT.CR = Desactivado) and (1er_paquete_CR_desactivado) and
                                    (DAT.SN ≠ SN_1er_paquete_CR_desactivado) THEN
        Fin_establecimiento = true
   ENDIF
ENDIF
RETURN
```

Suceso: ***Recibe_MTTUD***

```
                                    {Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (T_LIB , MAX_VAL_TLIB)
```

{Lo envía a sus hijos a través del árbol de distribución}

```
MTTUD.LRV = Identificador_propio
FOR i IN n_interfaces DO
   MTTU.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
RETURN
```

Suceso: ***Recibe_TS***

{En este estado si se recibe un paquete TS sólo tiene sentido que éste tenga el flag NC (nuevo hijo) activado, lo cual significa que uno de sus hijos intenta unirse al árbol RMNP}

```
IF (flag NC activado) THEN
   Alta_INF_de_TIB(nuevo_hijo)          {Da de alta en la TIB al nuevo hijo}
   Alta_INF_Est(nuevo_hijo)
            {Guarda la información del hijo y queda pendiente hasta que él mismo no reciba del padre la confirmación de su pertenencia al árbol}
   PUT (PILA_TS_ACK_pendientes, Info_hijo_pendiente)
ELSE
   Descarta_paquete
ENDIF
RETURN
```

Suceso: ***Recibe_TS_ACK***

{El padre le confirma que pertenece ya al árbol de distribución RMNP, y además le da la información de estado: identificador del último reinicio, si éste se encuentra activo o no, el último paquete confirmado por el padre y el próximo paquete esperado}

```
IF (flag NC está activado) and (está activado T_WF) THEN
   RESET(T_WF)
   Identificador_último_reinicio = TS_ACK.RI
   Estado_reinicio = TS_ACK.RS
   Último_confirmado = TS_ACK.LC
            {Envía a cada hijo que estaba pendiente de entrar en el árbol el correspondiente TS_ACK}
   WHILE NOT (Vacia (PILA_TS_ACK_pendientes)) DO
         Info_hijo_pendiente = GET (PILA_TS_ACK_pendientes)
         TS_ACK.NC = Activado
         TS_ACK.RI = Identificador_último_reinicio     {Informa del último reinicio,}
         TS_ACK.RS = Estado_reinicio     {Informa del estado del último reinicio}
                                         {Indica cual es el último paquete confirmado}
         TS_ACK.LC = Último_confirmado
```

```
            Envía_TS_ACK(hijo_pendiente)        {Confirma al hijo su pertenencia al árbol}
        END_WHILE
        IF (es un SR) THEN reserva_buffers ENDIF
        IF (Estado_reinicio = Activado) THEN
            ESTADO (Reinicio)
        ELSE_IF (Fin_establecimiento)
            ESTADO (Normal)
          ELSE
            ESTADO (Establecimiento)
          ENDIF
        ENDIF
    ELSE
        Error
    ENDIF
    RETURN
```

Suceso: ***Recibe_LIB***

{Distribuye el paquete y libera recursos}

```
    FOR i IN n_interfaces DO
        Difunde_paquete_X_interfaz[i]
    END_FOR
    Libera_recursos
    ESTADO (Fin)
```

Suceso: ***Expira_T_{LIB}***

{Asume que se ha producido una liberación de la conexión y deja los recursos locales asociados a ésta}

```
    Libera_recursos
    ESTADO (Fin)
```

Suceso: ***Expira_T_{WF}***

{Reenvía el paquete TS con el campo NC activado, flag de nuevo hijo, al padre}

```
    IF (num_retransmisiones_a_padre= MAX_TWF) THEN
        Libera_recursos
        ESTADO (Fin)
    ENDIF
    Incrementa_num_retransmisiones_a_padre
    TS.NC = Activado
    Envía_TS(padre)                    {Reenvía el TS que estaba asociado al T_WF que expiró}
    Calcula_T_WF
```

Activa_T_{WF} {Activa el temporizador de espera TS_ACK}
RETURN

Estado Establecimiento

{Difunde todos los paquetes que le envía la fuente, sin realizar control intermedio de secuencia. No generan NAKs ni realizan filtrado de retransmisiones. Los SR se comportan de igual manera que los BRs y no realizan funciones de almacenamiento ni de retransmisión. Finaliza la fase de establecimiento cuando se reciben dos paquetes de datos con el flag CR desactivado, y entonces transita al estado de Distribución Normal}

```
BUCLE_INFINITO
   WHILE NOT(existe suceso) DO
        CALL Proceso_ocioso
   END_WHILE
     CASE suceso IN
        Recibe_DAT:          GOSUB(Recibe_DAT)
        Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
        Recibe_MTTUD_ACK:    GOSUB(Recibe_MTTUD_ACK)
        Recibe_TS:           GOSUB(Recibe_TS)
        Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
        Recibe_LIB:          GOSUB(Recibe_LIB)
        Expira_T_LIB:        GOSUB(Expira_T_LIB)
        Expira_T_WF:         GOSUB(Expira_T_WF)
        Expira_T_WS:         GOSUB(Expira_T_WS)
     ESAC
FIN_BUCLE
```

Suceso: *Recibe_DAT*

{Difunde todos los paquetes que le envía la fuente, y trata de detectar que ha finalizado la fase de establecimiento al recibir un segundo paquete con el CR desactivado}

{Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (T_{LIB} , MAX_TLIB)

{Comprueba si se ha iniciado un proceso de reinicio}
CALL_MOD (Comprueba_reinicio)

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}
CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que le envía el padre los paquetes de datos. Si es así avisará a su padre}

```
CALL_MOD (Comprueba_interfaz)
                                        {Distribuye el paquete por todos los interfaces}
DAT.LRV = Identificador_propio
FOR i IN n_interfaces DO
   DAT.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
                                        {Comprueba si es el primero con el CR desactivado}
IF (DAT.CR = Desactivado) and NOT(1er_paquete_CR_desactivado) THEN
        1er_paquete_CR_desactivado = true
            {Comprueba si es un segundo paquete recibido con CR desactivado, y si ambos paquetes son distintos}
ELSE_IF (DAT.CR = Desactivado) and (1er_paquete_CR_desactivado) and
                                    (DAT.SN ≠ SN_1er_paquete_CR_desactivado) THEN
        ESTADO (Distribución Normal)
   ENDIF
ENDIF
RETURN
```

Suceso: ***Recibe_MTTUD***

{Lo envía a sus hijos a través del árbol de distribución}

```
                                        {Actualiza el temporizador de liberación por silencio de la fuente}
Actualiza (T_LIB , MAX_TLIB)
                                        {Comprueba si se ha iniciado un proceso de reinicio}
CALL_MOD (Comprueba_reinicio)
            {Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un reinicio}
CALL_MOD (Comprueba_cambio_árbol)
            {Comprueba si ha variado el interfaz por el que le envía el padre los paquetes de datos. Si es así avisará a su padre}
CALL_MOD (Comprueba_interfaz)
                                        {Distribuye el paquete por todos los interfaces}
MTTUD.LRV = Identificador_propio
FOR i IN n_interfaces DO
   MTTU.I = i
   Difunde_paquete_X_interfaz[i]
END_FOR
RETURN
```

Suceso: ***Recibe_MTTUD_ACK***

{Realiza el proceso de agregación de MTTUD}

```
Actualiza_INF_Est(hijo)
IF NOT(existe hijo por confirmar MTTUD) THEN
   Envía_MTTUD_ACK          {Envía confirmación del paquete MTTUD al padre}
ENDIF
RETURN
```

Suceso: ***Recibe_TS***

```
CASE (flag activo) IN
                                        {Un hijo informa de un cambio de interfaz}
   NI:  Modificación_INF_de_TIB(hijo)
        TS_ACK.NI = Activado
        Envía_TS_ACK(hijo)
                                                        {Se incorpora un miembro}
   NC:  Alta_INF_de_TIB(nuevo_hijo)          {Da de alta en la TIB al nuevo hijo}
        Alta_INF_Est(nuevo_hijo)             {Actualiza su información de estado}
        TS_ACK.NC = Activado
        TS_ACK.RI = Último_RI                        {Informa del último reinicio}
        TS_ACK.RS = Estado_RI          {Informa del estado del último reinicio}
        TS_ACK.LC = Ultimo_confirmado
        Envía_TS_ACK(nuevo_hijo)
                                                {Un hijo solicita salir del árbol RMNP}
   ML:  Baja_INF_de_TIB(hijo)
        Baja_INF_Est(hijo)
        TS_ACK.ML = Activado
        Envía_TS_ACK(hijo)
                                        {Si no tiene hijos solicita salir del árbol RMNP}
        IF (número_hijos = 0) THEN
              TS.ML = Activado
              Envía_TS(padre)                {Solicita a su padre la desconexión}
              IF (es un SR) THEN Libera_Buffers ENDIF
              FOR t IN t_temporizadores DO          {Detiene los temporizadores}
                    RESET (t)
              END_FOR
              Calcula_T_WF
              Activa_T_WF
              ESTADO (Salida Árbol Solicitada)
        ENDIF
ESAC
```

RETURN

Suceso: ***Recibe_TS_ACK***

{Este suceso sólo es posible, si se ha producido un cambio en el interfaz del padre}

IF (flag NI está activado) and (está activado T_{WF}) THEN RESET(T_{WF})
ELSE Error ENDIF
RETURN

Suceso: ***Recibe_LIB***

{En este estado si ocurre una liberación será un aborto de la conexión}

```
FOR i IN n_interfaces DO
   Difunde_paquete_X_interfaz[i]
END_FOR
IF (LIB.AB está activo) THEN
        Libera_recursos
        ESTADO (Fin)
ELSE
   Error
ENDIF
```

Suceso: ***Expira_TLIB***

{Interrumpe el proceso de establecimiento, y el encaminador asume que se ha producido una liberación de la conexión y deja los recursos locales asociados a ésta}

Libera_recursos
ESTADO (Fin)

Suceso: ***Expira_TWF***

{Expira el temporizador de espera del TS_ACK del padre. Esta situación sólo puede deberse a que ha habido un cambio de interfaz en el padre}

```
                                {Si es alcanzado el MAX_TWF se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWF)
TS.NI = Activado                                {Flag de nuevo interfaz activado}
Indica_nuevo_interfaz_en_TS
Incrementa_num_retransmisiones_a_padre
Envía_TS(padre)
Calcula_TWF
Activa_TWF                          {Activa el temporizador de espera TS_ACK}
RETURN
```

Suceso: ***Expira_TWS***

{Expira el temporizador de espera de confirmación de la fuente asociado al paquete PAQ}

{Si es alcanzado el MAX_TWS se realiza una liberación parcial}
CALL_MOD (Comprueba_MAX_TWS)
Incrementa_num_retransmisiones_a_fuente
Envía_PAQ(fuente) {Reenvía el paquete que estaba asociado al T_{WS} que expiró}
Calcula_T_{WS}
Activa_T_{WS} {Activa el temporizador de espera TS_ACK}
RETURN

Estado Reinicio

{A este estado se transita cuando ocurre un cambio en el árbol RMNP}

```
BUCLE_INFINITO
  WHILE NOT(existe suceso) DO
      CALL Proceso_ocioso
  END_WHILE
    CASE suceso IN
      Recibe_DAT:          GOSUB(Recibe_DAT)
      Recibe_MTTUD         GOSUB(Recibe_MTTUD)
      Recibe_MTTUD_ACK     GOSUB(Recibe_MTTUD_ACK)
      Recibe_ACK:          GOSUB(Recibe_ACK)
      Recibe_TS:           GOSUB(Recibe_TS)
      Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
      Recibe_LIB:          GOSUB(Recibe_LIB)
      Recibe_LIB_ACK:      GOSUB(Recibe_LIB_ACK)
      Expira_T_LIB:        GOSUB(Expira_T_LIB)
      Expira_T_WF:         GOSUB(Expira_T_WF)
      Expira_T_WS:         GOSUB(Expira_T_WS)
    ESAC
FIN_BUCLE
```

Suceso: ***Recibe_DAT***

Actualiza (T_{LIB} , MAX_VAL_TLIB)

{Comprueba si ha variado el padre que le envía datos. Si es así, solicita a la fuente un nuevo reinicio}

CALL_MOD (Comprueba_cambio_árbol)

{Comprueba si ha variado el interfaz por el que su padre le envía paquetes de datos. Si es así avisará a éste}

```
CALL_MOD (Comprueba_interfaz)
                    {Comprueba si el paquete es anterior al procedimiento de reinicio actual}
IF (DAT.RI < Último_RI) OR (DAT.RI = 0) THEN
   Descarta_paquete
ELSE                                    {Distribuye el paquete por todos los interfaces}
   DAT.LRV = Identificador_propio
   FOR i IN n_interfaces DO
         DAT.I = i
         Difunde_paquete_X_interfaz[i]
   END_FOR
ENDIF
                                  {Si ha finalizado el reinicio pasa a Distribución Normal}
IF (DAT.RS = 0 and DAT.RI=Ultimo.RI) THEN
   FOR i IN n_hijos DO
         IF NOT(comprobada_INF_de_TIB(hijo)) THEN
               Borra_INF_de_TIB(hijo)
               Borra_INF_Est(hijo)
         ENDIF
   END_DO
   ESTADO (Distribución Normal)
            {Comprueba si la fuente a puesto en marcha un nuevo procedimiento de reinicio}
  ELSE_IF (DAT.RI > Último_RI) THEN
   Último_RI = DAT.RI                   {Actualiza_información_último_reinicio}
   Actualiza_INF_de_TIB(padre)
   TS.RI = Último_RI                {Avisa a su padre de que continúa siendo su hijo}
   TS.TC = Activado
   Envía_TS(padre)
   Calcula_T_WF
   Activa_T_WF
            {Comprueba si estaba pendiente de que la fuente pusiera en marcha un reini-
            cio y si es así detiene el temporizador asociado}
   IF (reinicio_pendiente) and (está activado T_WS) THEN
         RESET(T_WS)
   ENDIF
  ENDIF
ENDIF
RETURN
```

Suceso: ***Recibe_MTTUD***

{Al ponerse en marcha un reinicio la fuente detiene todo proceso de descubrimiento de MTTU}

Descarta_paquete
RETURN

Suceso: ***Recibe_MTTUD_ACK***

{Al ponerse en marcha un reinicio la fuente detiene todo proceso de descubrimiento de MTTU}

Descarta_paquete
RETURN

Suceso: ***Recibe_ACK***

{Actualiza cual será el paquete esperado, para cuando finalice el proceso de reinicio}

paquete esperado = ACK.SN+1
IF NOT (reinicio suave) THEN Actualiza_INF_Est(hijo) ENDIF
RETURN

Suceso: ***Recibe_TS***

CASE (flag activo) IN

{Activado el flag de comprobación del árbol. Actualiza y marca como comprobada la entrada en la TIB relativa al correspondiente hijo}

TC: Actualiza_INF_de_TIB(hijo)
TS_ACK.RI = Último_RI {Envía confirmación al hijo}
TS_ACK.RS = Activado
TS_ACK.TC = Activado
Envía_TS_ACK(hijo)

{Un hijo informa de un cambio de interfaz}

NI: Modificación_INF_de_TIB(hijo)
TS_ACK.NI = Activado
Envía_TS_ACK(hijo)

{Se incorpora un miembro}

NC: IF (Reinicio_suave) THEN
Alta_INF_de_TIB(nuevo_hijo)
Alta_INF_Est(nuevo_hijo)
TS_ACK.NC = Activado
TS_ACK.RI = Último_RI {Informa del último reinicio}
TS_ACK.RS = Estado_RI {Informa del estado del último reinicio}
TS_ACK.LC = Último_confirmado
Envía_TS_ACK(nuevo_hijo)
ELSE Descarta_paquete ENDIF

{Un hijo solicita salir del árbol RMNP}

ML: Baja_INF_de_TIB(hijo)
Baja_INF_Est(hijo)

```
            TS_ACK.ML = Activado
            Envía_TS_ACK(hijo)
                                        {Si no tiene hijos solicita salir del árbol RMNP}
            IF (número_hijos = 0) THEN
                  TS.ML = Activado
                  Envía_TS(padre)              {Solicita a su padre la desconexión}
                  FOR t IN t_temporizadores DO        {Detiene los temporizadores}
                        RESET (t)
                  END_FOR
                  Calcula_T_WF
                  Activa_T_WF
                  ESTADO (Salida Árbol Solicitada)
            ENDIF
      ESAC
      RETURN
```

Suceso: ***Recibe_TS_ACK***

{El padre confirma que comprobó la información en la TIB o que actualizó el interfaz}

```
      IF ((flag NI activo) or (flag TC activo))and (está activado T_WF) THEN
         RESET (T_WF)
      ELSE
            Error
      ENDIF
      RETURN
```

Suceso: ***Recibe_LIB***

{Se ha producido una liberación ordenada o aborto. Una liberación ordenada sólo se puede dar si el reinicio es suave}

```
      FOR i IN n_interfaces DO
         Difunde_paquete_X_interfaz[i]
      END_FOR
                              {El comportamiento depende de como sea el tipo de liberación}
      IF (LIB.OR está activo) THEN
         FOR t IN t_temporizadores DO
            IF NOT(t igual T_LIB) THEN RESET (t) ENDIF
         END_FOR
         ESTADO (Liberación Ordenada)
      ELSE_IF (LIB.AB está activo) THEN
            Libera_recursos
            ESTADO (Fin)
```

```
    ENDIF
  ENDIF
  RETURN
```

Suceso: ***Recibe_LIB_ACK***

{En este estado la única posibilidad es que este LIB_ACK lleve el flag INF activado y haya sido enviado por la fuente para confirmar que recibió el aviso de que se ha producido una liberación parcial}

```
  IF (flag INF está activado) and (está activado T_WS) THEN RESET(T_WS)
  ELSE Error ENDIF
  RETURN
```

Suceso: ***Expira_T_LIB***

{El encaminador asume que se ha producido una liberación de la conexión y libera los recursos locales asociados a ésta}

```
  Libera_recursos
  ESTADO (Fin)
```

Suceso: ***Expira_T_WF***

{El temporizador estará asociado a un paquete TS, que puede solicitar o una comprobación del árbol o un cambio de interfaz. En cualquiera de los casos reenvía el paquete TS y reinicia el temporizador T_{WF}}

```
                          {Si es alcanzado el MAX_TWF se realiza una liberación parcial}
  CALL_MOD (Comprueba_MAX_TWF)
  Incrementa_num_retransmisiones_a_padre
  Envía_TS(padre)          {Reenvía el TS que estaba asociado al T_WF que expiró}
  Calcula_T_WF
  Activa_T_WF                          {Activa el temporizador de espera TS_ACK}
  RETURN
```

Suceso: ***Expira_T_WS***

{Expira el temporizador de espera de confirmación de la fuente asociado al paquete PAQ. Este paquete asociado al temporizador puede ser un TS con el flag RN activado (avisar a la fuente de que se ha producido un nuevo reinicio) o un LIB con el flag INF activado (informar de una liberación parcial)}

```
  CALL_MOD (Comprueba_MAX_TWS)
  Incrementa_num_retransmisiones_a_fuente
  Envía_PAQ(fuente)        {Reenvía el paquete que estaba asociado al T_WS que expiró}
  Calcula_T_WS
```

Activa_T_{WS} {Activa el temporizador de espera TS_ACK}
RETURN

Estado Salida Árbol Solicitada

{Estado al que se llega por haberse dado de baja el último hijo al que se estaba dando servicio}

```
BUCLE_INFINITO
  WHILE NOT(existe suceso) DO
      CALL Proceso_ocioso
  END_WHILE
    CASE suceso IN
      Recibe_DAT:        GOSUB(Recibe_DAT)
      Recibe_MTTUD       GOSUB(Recibe_MTTUD)
      Recibe_TS_ACK:     GOSUB(Recibe_TS_ACK)
      Recibe_LIB:        GOSUB(Recibe_LIB)
      Expira_TWF:        GOSUB(Expira_TWF)
    ESAC
FIN_BUCLE
```

Suceso: ***Recibe_DAT, Recibe_MTTUD***

Descarta_paquete
RETURN

Suceso: ***Recibe_TS_ACK***

Libera_recursos
ESTADO (Fin)

Suceso: ***Recibe_LIB***

```
IF (LIB.OR está activo) THEN
  Descarta_paquete
ELSE_IF (LIB.AB está activo) THEN
      Libera_recursos
      ESTADO (Fin)
   ENDIF
ENDIF
RETURN
```

Suceso: ***Expira_TWF***

```
IF (num_retransmisiones_a_padre= MAX_TWF) THEN
  Libera_recursos
  ESTADO (Fin)
```

```
ENDIF
Incrementa_num_retransmisiones_a_padre
TS.ML = Activado
Envía_TS(padre)                    {Solicita de nuevo a su padre la desconexión}
Calcula_T_WF
Activa_T_WF                        {Activa el temporizador de espera TS_ACK}
RETURN
```

Estado Avisando Liberación Parcial

{Estado al que se pasa cuando un encaminador RMNP tras realizar una liberación parcial, se ha quedado sin hijos y está esperando a que la fuente confirme que recibió el correspondiente aviso con un LIB_ACK. Tras recibir dicha confirmación, él mismo solicitará salir del árbol RMNP}

```
BUCLE_INFINITO
  WHILE NOT(existe suceso) DO
      CALL Proceso_ocioso
  END_WHILE
    CASE suceso IN
      Recibe_DAT:          GOSUB(Recibe_DAT)
      Recibe_MTTUD:        GOSUB(Recibe_MTTUD)
      Recibe_MTTUD_ACK:    GOSUB(Recibe_MTTUD_ACK)
      Recibe_ACK:          GOSUB(Recibe_ACK)
      Recibe_NAK:          GOSUB(Recibe_NAK)
      Recibe_TS:           GOSUB(Recibe_TS)
      Recibe_TS_ACK:       GOSUB(Recibe_TS_ACK)
      Recibe_LIB:          GOSUB(Recibe_LIB)
      Recibe_LIB_ACK:      GOSUB(Recibe_LIB_ACK)
      Expira_T_LIB:        GOSUB(Expira_T_LIB)
      Expira_T_WF:         GOSUB(Expira_T_WF)
      Expira_T_WS:         GOSUB(Expira_T_WS)
    ESAC
FIN_BUCLE
```

Suceso: ***Recibe_DAT***

{Confirma el paquete recibido y descarta el paquete}

```
Actualiza (T_LIB , MAX_TLIB)
ACK.SN = DAT.SN
Envía_ACK(padre)
Descarta_paquete
RETURN
```

Suceso: ***Recibe MTTUD***

Actualiza (T_{LIB} , MAX_TLIB)
Envía_MTTUD_ACK(padre)
Descarta_paquete
RETURN

Suceso: ***Recibe_ACK o Recibe_NAK o Recibe_TS o Recibe_MTTUD_ACK***

{Estos paquetes pueden haber sido enviados por los hijos antes de recibir el paquete de liberación}

Descarta_paquete
RETURN

Suceso: ***Recibe_TS_ACK***

{Sólo es posible este suceso, si se había producido un cambio en el interfaz antes de la liberación parcial}

RESET(T_{WF}) {Detiene temporizador}
RETURN

Suceso: ***Recibe_LIB***

{El comportamiento depende de qué flag se encuentra activo}

```
IF (LIB.OR está activo) THEN
   Descarta_paquete
ELSE_IF (LIB.AB está activo) THEN
         Libera_recursos
         ESTADO (Fin)
      ENDIF
ENDIF
RETURN
```

Suceso: ***Recibe_LIB_ACK***

{La fuente confirma que ha recibido el aviso de que se ha producido una liberación parcial}

```
IF (LIB_ACK.INF está activado) and (está activo T_WS) THEN
   RESET(T_WS)
            {El mismo sale del árbol}
   TS.ML = Activado
   Envía_TS(padre)
   Calcula_T_WF
   Activa_T_WF
```

RESET (T_{LIB})
ESTADO (Salida Árbol Solicitada)
ELSE
Error
ENDIF
RETURN

Suceso: ***Expira_TLIB***

Libera_recursos
ESTADO (Fin)

Suceso: ***Expira_TWF***

{Expira el temporizador de espera del TS_ACK del padre sólo cuando hay un cambio de interfaz}

IF (num_retransmisiones_a_padre= MAX_TWF) THEN
Libera_recursos
ESTADO (Fin)
ENDIF
Incrementa_num_retransmisiones_a_padre
Envía_TS(padre) {Reenvía el TS que estaba asociado al T_{WF} que expiró}
Calcula_T_{WF}
Activa_T_{WF} {Activa el temporizador de espera TS_ACK}
RETURN

Suceso: ***Expira_TWS***

{Expira el temporizador de espera de confirmación de la fuente asociado al paquete PAQ, el sistema reenvía dicho paquete}

IF (num_retransmisiones_a_fuente= MAX_TWS) THEN
Libera_recursos
ESTADO (Fin)
ENDIF
Incrementa_num_retransmisiones_a_fuente
Envía_PAQ(fuente) {Reenvía el paquete que estaba asociado al T_{WS} que expiró}
Calcula_T_{WS}
Activa_T_{WS} {Activa el temporizador de espera TS_ACK}
RETURN

Estado Liberación Ordenada

{El sistema ha recibido un paquete de la fuente solicitando una liberación ordenada de la conexión, y después de distribuir dicho paquete a sus hijos se encuentra esperando los correspondientes LIB_ACK para realizar el proceso de agregación de éstos}

```
BUCLE_INFINITO
   WHILE NOT(existe suceso) DO
         CALL Proceso_ocioso
   END_WHILE
     CASE suceso IN
         Recibe_TS:          GOSUB(Recibe_TS)
         Recibe_TS_ACK:      GOSUB(Recibe_TS_ACK)
         Recibe_LIB:         GOSUB(Recibe_LIB)
         Recibe_LIB_ACK:     GOSUB(Recibe_LIB_ACK)
         Expira_T_LIB:       GOSUB(Expira_T_LIB)
      ESAC
FIN_BUCLE
```

Suceso: ***Recibe_TS***

{Pueden haber sido enviados por un hijo antes de recibir el paquete de liberación}

```
Descarta_paquete
RETURN
```

Suceso: ***Recibe_TS_ACK***

{Sólo es posible este suceso, si se ha producido un cambio en el interfaz}

```
RESET(T_WF)          {Detiene temporizador}
RETURN
```

Suceso: ***Recibe_LIB***

{Sólo tiene sentido que tenga el flag OR activado}

```
IF (todo hijo envió LIB_ACK) THEN
   LIB_ACK.OR = Activado
   Envía_LIB_ACK(padre)
ELSE
   FOR i IN n_interfaces DO
         Difunde_paquete_X_interfaz[i]
   END_FOR
ENDIF
RETURN
```

Suceso: ***Recibe_LIB_ACK***

{Realiza un proceso de agregación de paquetes LIB_ACK, y si es un SR libera buffers}

Actualiza_INF_Est(hijo)
IF (todo hijo envió LIB_ACK) THEN
  LIB_ACK.OR = Activado
  Envía_LIB_ACK(padre)
ENDIF
RETURN

Suceso: ***Expira_T_{LIB}***

{El encaminador deja libres los recursos locales asociados a la conexión RMNP}

Libera_recursos
ESTADO (Fin)

Estado Fin

{.}

LISTA DE ACRÓNIMOS

- *AFDF: Adaptive File Distribution Protocol*
- *ATM: Asynchronous Transfer Mode*
- BR: Encaminador RMNP con funcionalidad básica
- *CBT: Core Based Tree*
- *CCITT: Comité Consultatif Internationale Télégraphique et Téléphonique*
- *CLNP: ConnectionLess-mode Network Protocol*
- *CSCW: Computer Supported Cooperative Work*
- *DIS: Distributed Interactive Simulation*
- *DVMRP: Distance-Vector Multicast Routing Protocol*
- *FTM: Fault-Tolerance internet Multicasting*
- *ICMP: Internet Control Message Protocol*
- *IETF: Internet Engineering Task Force*
- *IGMP: Internet Group Management Protocol*
- *IP: Internetwork Protocol*
- *ISO: International Standardization Organization*
- *ITU-T: International Telecommunication Union - Telecommunication Standardization Sector*
- *LBRM: Log-Based Receiver_reliable Multicast*
- *LLC: Logical Link Control*
- *MAC: Media Access Control*
- *MBone: Internet Multicast Backbone*
- *MOSPF: Multicast Open Shortest-Path First*

- *MTP: Multicast Transport Protocol*
- MTTU: Mínima MTU en el árbol de distribución
- *MTU: Maximun Transfer Unit*
- *OSI: Open Systems Interconnection*
- *OSPF: Open Shortest-Path First*
- *PIM: Protocol Independent Multicast*
- *QoS: Quality of Service*
- *RAL: Red de Area Local*
- *RFC: Request For Comments*
- *RIP: Routing Information Protocol*
- RMNP: Reliable Multicast Network Protocol
- *RMTP: Reliable Multicast Transport Protocol*
- *RMP: Reliable Multicast Protocol*
- *RTT: Round Trip Time*
- *SDU: Service Data Unit*
- *SMDS: Switched Multimegabit Data Service*
- *SRM: Scalable Reliable Multicast*
- SR: Encaminador RMNP con capacidad de almacenamiento y retransmisión
- IRTT: Tiempo de ida y vuelta por un interfaz
- *TCP: Transmission Control Protocol*
- TIB: Base de información de árboles RMNP
- *TMTP: Tree-based Multicast Transport Protocol*
- *TTL: Time To Live*
- *WAN: Wide Area Network*
- *XTP: Xpress Transport Protocol*
- *XTPX: Xpress Transport Protocol eXtended*

BIBLIOGRAFÍA

[AC95] A. Azcorra and M. Calderón. A Network-Based Approach to Reliable Multicast. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, pp 393-404, October 1995.

[AE92] S.R. Ahuja and R. Ensor. Coordination and Control of Multimedia Conferencing. *IEEE Communications Magazine*, pp. 38-43, May 1992.

[AFM92] S. Armstrong, A. Freier and K. Marzullo. Multicast Transport Protocol. *RFC 1301*, February 1992.

[APR93] R. Aiello, E. Pagani and G.P. Rossi. Causal Ordering in Reliable Group Communications. *Proc. of ACM SIGCOMM'93*, 23(4):106-115, September 1993.

[ASDM91] C. Alaettinoglu, U. Shankar, K. Dussa-Zieger and I. Matta. MaRS (Maryland Routing Simulator) - version 1.0 user's manual. *Technical Report, CS-TR-2687, Department of Computer Science, University of Maryland*, June 1991.

[ASDM92] C. Alaettinoglu, U. Shankar, K. Dussa-Zieger and I. Matta. Design and Implementation of MaRS: A Routing Tesbed. *Technical Report, CS-TR-2964, Department of Computer Science, University of Maryland*, September 1992.

[AW92] M.H. Ammar and L-R. Wu. Improving the Performace of Point-to-Multipoint ARQ Protocols Through Destination Set Splitting. *Proc. of IEEE INFOCOM'92*, pp. 262-271, May 1992.

[Bal95] T. Ballardie. *A New Aproach to Multicasting Communication in a Datagram Internetwork*. PhD Thesis, University of London, May 1995.

[Bal96] T. Ballardie. Core Based Trees (CBT) Multicast Architecture. *Internet Working Draft*, February 1996.

[BC91] K. Birman and R. Cooper. The ISIS Project: Real Experience with a Fault Tolerant Programming System. *ACM Operating System Review*, 25(2):103-107, April, 1991.

[BC94] K. Birman and T. Clark. Performance of the ISIS Distributed Computing Toolkit. *Technical Report, TR-94-1432, Dept. of Computer Science, Cornell University*, June 1994.

[BCDB95] M. Borden, E. Crawley, B. Davie and S. Batsell. Integration of Real-Time Services in an IP-ATM Network Architecture. *RFC 1821*, May 1995.

[BCG91] K. Birman, R. Cooper and B. Gleeson. Programming with Process Groups: Group and Multicast Semantics. *Technical Report, TR-91-1185, Cornell University*, January 1991.

[BCS94] R. Braden, D. Clark and S. Shenker. Integrated Services in the Internet Architecture: An Overview. *RFC 1633*, June 1994.

[BFC93] T. Ballardie, P. Francis and J. Crowcroft. Core Based Trees (CBT) - An Architecture for Scalable Inter-Domain Multicast Routing. *Proc. of ACM SIGCOMM'93*, 23(4):85-95, September 1993.

[BG85] W. Bux and D. Grillo. Flow Control in Local-Area Networks of Interconnected Token Rings. *IEEE Transactions on Communications*, 33(10):1058-1066, October 1985.

[Bir93] K. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of the ACM*, 36(12):37-53, December 1993.

[BJ83] K. Bharath-Kumar and J.M. Jaffe. Routing to Multiple Destinations in Computer Networks. *IEEE Transactions on Communications*, 31(3):343-351, March 1983.

[BJ87] K. Birman and T.A. Joseph. Reliable Communication in Presence of Failures. *ACM Transactions on Computer Systems*, 5(1):47-76, February 1987.

[Böc92] S. Böcking. TEMPO: A Lightweight Transport Protocol. *Proc. of the third Workshop Future Trends of Distributed Computing Systems*, pp. 107-113, April, 1992.

[BOG+94] C. Bormann, J. Ott, H-C. Gehrcke, T. Kerschat and N. Seifert. MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport. *Proc. of ICCCN'94*, September 1994.

[BRJ96] T. Ballardie, S. Reeve and N. Jain. Core Based Trees (CBT) - Protocol Specification -. *Internet Working Draft*, February 1996.

[BSS91] K. Birman, A. Schiper and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, 9(3):272-314, August 1991.

[BTW94] J-C. Bolot, T. Turletti and I. Wakeman, Scalable Feedback Control for Multicast Video Distribution in the Internet. *Proc. of ACM SIGCOMM'94*, 24(4):58-67, September 1994.

[BZ93] R. Braudes and S. Zabele. Requirements for Multicast Protocols. *RFC 1458*. May 1993.

[CA94] S.Y. Cheung and M.H. Ammar. Using Destination Set Grouping to Improve the Performance of Window-Controlled Multipoint Connections. *Technical Report GIT-CC-94-32, Georgia Institute of Technology*, August 1994.

[CA95] R. Clark and M.H. Ammar. Providing Scalable Web Service Using Multicast Delivery. *Technical Report GIT-CC-95-03, Georgia Institute of Technology*, January 1995. (También en Proc. of IEEE workshop on Services in Distributed and Networked Environments, June 1995)

[CCI88] CCITT. Recommendation X.25: Interface between Data Terminal Equipment (DTE) and Data Communicating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks. *Blue Book*, 1988.

[CD92] S. Casner and S. Deering. First IETF Internet Audiocast. *ACM Computer Communication Review*, 22(3):92-97, July 1992.

[Cha84] J.M. Chang. Simplifying Distributed Database Systems Design by Using a Broadcast Network. *Proc. of SIGMOD'84*, pp. 223-233, June 1984.

[CJ89] D-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for Congestion avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1-14, 1989.

[CK95] J.R. Cooperstock and S. Kotsopoulos. Exploiting Group Communication for Reliable High Volume Data Distribution. *Proc. of IEEE PACRIM, Pacific RIM Conference on Communicationes, Computers, Visualization and Signal Processing*, May 1995.

[CK96] J.R. Cooperstock and S. Kotsopoulos. Why Use a Fishing Line When You Have a Net? An Adaptive Multicast Data Distribution Protocol. *Proc. of USENIX Technical Conference,* January 1996.

[Cla82] D.D. Clark. Window and Acknowledgement Strategy in TCP. *RFC 813*, July 1982.

[Cla88] D.D. Clark. The Design Philosophy of the Darpa Internet Protocols. *Proc of ACM SIGCOMM'88*, 18(4):106-114, August 1988.

[CM84] J.M. Chang and N.F. Maxemchuk. Reliable Broadcast Protocols. *ACM Transactions on Computer Systems*, 2(3):251-273, August 1984.

[Com91] D. Comer. *Internetworking with TCP/IP. Volume I; Principles, Protocols and Architecture*. Prentice-Hall International, 1991.

[CP88] J. Crowcroft and K. Paliwoda. A Multicast Transport Protocol. *Proc. of ACM SIGCOMM'88*, 18(4):247-256, August 1988.

[CRKG89] C. Cheng, R. Riley, S.P.R. Kumar and J.J. García-Luna Aceves. A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect. *Proc. of ACM SIGCOMM' 89*, 19(4):224-236, September 1989.

[CT90] D.D. Clark and D.L. Tennenhouse. Architectural Considerations For New Generation of Protocols. *Proc. of ACM SIGCOMM'90*, pp. 200-208, September 1990.

[CW89] D.R. Cheriton and C.L. Williamson. VMTP as the Transport Layer for High-Performance Distributed Systems. *IEEE Communications Magazine*, pp. 37-44, June 1989.

[CZ85] D.R. Cheriton and W. Zwaenepoel. Distributed Process Groups in the V Kernel. *ACM Transactions on Computer Systems*, 3(2):77-107, May 1985.

[Dan89] P.B. Danzing. Finite Buffers and Fast Multicast. *Performance Evaluation Review*, 17(1):108-117, May 1989.

[DB95] L. Delgrossi and L. Berger. Internet Stream Protocol Version 2 (ST2), Protocol Specification - Version ST2+. *RFC 1819*. August 1995.

[DC90] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85-110, May 1990.

[Dee89] S. Deering. Host Extensions for IP multicast. *RFC 1112*, August 1989.

[Dee91] S. Deering. *Multicast Routing in a Datagram Internetwork.* PhD Thesis, Stanford University, December 1991.

[DEF+94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu and L. Wei. An Architecture for Wide-Area Multicast Routing. *Proc. of ACM SIGCOMM'94*, 24(4):126-135, September 1994.

[DEF+96] S. Deering, D. Estrin, D. Farinacci, V. Jacobson,. C. Liu and L. Wei. Protocol Independent Multicast-Sparce Mode (PIM-SM): Protocol Specification. *Internet Working Draft*, February 1996.

[DH95] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6), Specification. *RFC 1883*, December 1995.

[DKS89] A. Demers, S. Keshav and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Proc. of ACM SIGCOMM'89*, pp. 13-24, September 1989.

[DLW94] B.J. Dempsey, M.T. Lucas and A.C. Weaver. Design and Implementation of a High Quality Video Distribution System using XTP Reliable Multicast. *Second International Workshop on Advanced Communications and Applications for High-Speed Networks*, September 1994.

[DM78] Y.K. Dalal and R.M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of de ACM*, 21(12):1040-1047, December 1978.

[Don95] J.E. Donnelley. WWW Media Distribution via Hopwise Reliable Multicast. *Proc. of Third International Word-Wide Web Conference*, April 1995.

[EFJ+96] D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma and A. Helmy. Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification. *Internet Working Draft*, February 1996.

[EGR91] C.A. Ellis, S.J. Gibbs and G.L. Rein. Groupware, Some Issues and Experiences. *Communications of the ACM*, 34(1):38-58, January 1991.

[DC90] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85-110, May 1990.

[Dee89] S. Deering. Host Extensions for IP multicast. *RFC 1112*, August 1989.

[Dee91] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD Thesis, Stanford University, December 1991.

[DEF+94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu and L. Wei. An Architecture for Wide-Area Multicast Routing. *Proc. of ACM SIGCOMM'94*, 24(4):126-135, September 1994.

[DEF+96] S. Deering, D. Estrin, D. Farinacci, V. Jacobson,. C. Liu and L. Wei. Protocol Independent Multicast-Sparce Mode (PIM-SM): Protocol Specification. *Internet Working Draft*, February 1996.

[DH95] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6), Specification. *RFC 1883*, December 1995.

[DKS89] A. Demers, S. Keshav and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Proc. of ACM SIGCOMM'89*, pp. 13-24, September 1989.

[DLW94] B.J. Dempsey, M.T. Lucas and A.C. Weaver. Design and Implementation of a High Quality Video Distribution System using XTP Reliable Multicast. *Second International Workshop on Advanced Communications and Applications for High-Speed Networks*, September 1994.

[DM78] Y.K. Dalal and R.M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of de ACM*, 21(12):1040-1047, December 1978.

[Don95] J.E. Donnelley. WWW Media Distribution via Hopwise Reliable Multicast. *Proc. of Third International Word-Wide Web Conference*, April 1995.

[EFJ+96] D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma and A. Helmy. Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification. *Internet Working Draft*, February 1996.

[EGR91] C.A. Ellis, S.J. Gibbs and G.L. Rein. Groupware, Some Issues and Experiences. *Communications of the ACM*, 34(1):38-58, January 1991.

[Eri94] H. Eriksson. MBONE: The Multicast Backbone. *Communications of the ACM*, 37(8):54-60. August 1994.

[ES88] A. Erramilli and R.P. Singh. A Reliable and Efficient Multicast Protocol for Broadcast Networks. *ACM Computer Communication Review*, 17(5):343-352. August 1988.

[FBZ94] D. Ferrari, A. Banerjea and H. Zhang. Network Support for Multimedia - A discussion of the Tenet Approach. *Computer Networks and ISDN Systems*, 26(10):1267-1280, July 1994.

[FJM+95] S. Floyd, V. Jacobson, S. McCanne, C-G Liu and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *Proc. of ACM SIGCOMM'95*, August 1995. (También en ftp://ftp.ee.lbl.gov/papers/wb.tech.ps.Z)

[FM94] V. Frost and B. Melamed. Traffic Modeling for Telecommunications Networks. *IEEE Communications Magazine*, 32(3):70-81, March 1994.

[Gie92] M. Gien. From Operating Systems to Cooperative Operating Environments. *Technical Report, Chorus Systems, CS/TR-92-82*, October 1992.

[GS91] H. García-Molina and A. Spauster. Ordered and Reliable Multicast Communication. *ACM Transactions on Computer Systems*, 9(3):242-271, August 1991.

[Hed88] C. Hedrick. Routing Information Protocol. *RFC 1058*, June 1988.

[HS95] O. Hermanns and M. Schuba. Performance Investigations of the IP Multicast Architecture. *Proc. of JENC6*, May 1995.

[HSC95] H.W. Holbrook, S.K. Singhal and D.R. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. *Proc. of ACM SIGCOMM'95*, August 1995.

[HSE95] S. Herzog, S. Shenker and D. Estrin. Sharing the Cost of Multicast Trees: An Axiomatic Analysis. *Proc. of ACM SIGCOMM'95*, August. 1995.

[Hug88] L. Hughes. A Multicast Interface for UNIX 4.3. *Software-Practice and Experience*, 18(1):15-27, January 1988.

[HW95] M. Handley and I. Wakeman. CCCP: Conference Control Channel Protocol - A scalable base for building conference control applications (V1.4). Disponible en *http://hill.lut.ac.uk/DS-Archive/MTP.html*, 1995.

[ISIS] ISIS and HORUS www page, URL http://www.cs.cornell.edu/Info/Projects/ISIS/ISIS.html.

[ISO84] ISO. ISO 7498: Information Processing Systems Interconnection - Open Systems Interconnection - Basic Reference Model, October 1984.

[ITU93] ITU-T. Draft Recommendation T.122: Multipoint Communication Service for Audiographics and Audiovisual Conferencing Service Definition, 1993.

[ivs] T. Turletti. *INRIA Video Conferencing System.* Disponible en http://www.inria.fr/rodeo/personnel/Thierry.Turletti/ivs.html.

[Jac88] V. Jacobson. Congestion Avoidance and Control. *Proc. of SIGCOMM'88*, (18)4:314-329, September 1988.

[Jai86] R. Jain. A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks. *IEEE Journal on Selected Areas in Communications*, 4(7):1162-1167, October 1986.

[Jai89] R. Jain. A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks. *Computer Communications Review*, 19:56-71, October 1989.

[JBB92] V. Jacobson, R. Braden and D. Borman. TCP Extensions for High Performance. *RFC 1323*. May 1992.

[JSW91] M. Jones, S. Sorensen and R. Wilbur. Protocol Design for Large Group Multicasting: the Message Distribution Protocol. *Computer Communications*, 14(5):287-297, June 1991.

[Kad94] J. Kadirire. Minimising Packet Copies in Multicast Routing by Exploiting Geographic Spread. *ACM Computer Communication Review*, 24(3):47-62, July 1994.

[KIFU95] H. Kashima, Y. Ishida, Z. Furukawa and K. Ushijima. Searching Internet Resources using IP Multicast. http://inet.nttam.com. May 1995.

[KM87] C.A. Kent and J.C. Mogul. Fragmentation Considered Harmful. *Proc. of ACM SIGCOMM'87*, 17(5):390-401, October 1987.

[Kno93] S. Knowles. IESG Advice from Experience with Path MTU Discovery. *RFC 1435*, March 1993.

[KP87] P. Karn and C. Partridge. Improving Round-Trip Time estimates in Reliable Transport Protocols. *Proc. of SIGCOMM'87*, 17(5), October 1987.

[KTHB89] M.F. Kaashoek, A.S. Tanenbaum, S.F. Hummel and H.E. Bal. An Efficient Reliable Broadcast Protocol. *ACM Operating System Review*, 23(4):5-19, October 1989.

[Lam78] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of de ACM*, 21(7):558-564, July 1978.

[LB91] G. Le Lann and G. Bres. Reliable Atomic Broadcast in Distributed Systems with Omission Faults. *ACM Operating System Review*, 25(2):80-86, April 1991.

[LDW96] M.T. Lucas, B.J. Dempsey and A.C. Weaver. Distributed Error Recovery for Continuous Media Data in Wide-Area Multicast. Enviado a *IEEE INFOCOMM'96*.

[Lea93] C-T. Lea. A Multicast Broadband Packet Switch. *IEEE Transactions on communications*, 41(4):621-630, April 1993.

[LLSG92] R. Ladin, B. Liskov, L. Shrira and S. Ghemawat. Providing High Availability Using Lazy Replication. *ACM Transactions on Computer Systems*, 10(4):360-391, November 1992.

[LM94] A.M. Law and M.G. McComas. Simulation Software for Communications Networks: The State of the Art. *IEEE Communications Magazine*, 32(3):44-50, March 1994.

[LP91] J. Lawrence and D. Piscitello. The Transmission of IP Datagrams over the SMDS Service. *RFC 1209*, March 1991.

[LS96] J.C. Lin and P. Sanjoy. RMTP: A Reliable Multicast Transport Protocol. Por aparecer en *IEEE INFOCOMM'96*.

[LSH95] H. Linder, W. Stering and U. Hofmann. Performance of Multimedia Transport in LFNs. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, pp. 207-215. October 1995.

[Mar95] D. Marlow. Host Group Extensions for CLNP Multicasting. *RFC 1768*, March 1995.

[May92] E. Mayer. An Evaluation Framework for Multicasting Ordering Protocols. *Proc. of ACM SIGCOMM'92*, pp. 177-187, August 1992..

[MB94] M.R. Macedonia and D.P. Brutzman. Mbone Provides Audio and Video Across the Internet. *IEEE Computer*, 27(4):30-36, April 1994.

[MBM95] S. Maffeis, W. Bischofberger and K-U. Mätzel. A Generic Multicast Transport Service to Support Disconnected Operation. *Proc. of the 2nd USENIX Symposium on Mobile and Location-Independent Computing*, April 1995.

[MD90] J. Mogul and S. Deering. Path MTU Discovery. *RFC 1191*, November 1990.

[MK95] I. Miloucheva and T. Kerschat. Protocols for Efficient Multiparty Multimedia Applications. *Proc. of JENC6*, May 1995.

[MMA90] P.M. Melliar-Smith, L.E. Moser and V. Agrawala. Broadcast Protocols for Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):17-25, January 1990.

[Mon94] T. Montgomery. *Design, Implementation, and Verification of the Reliable Multicast Protocol.* Master's Thesis, West Virginia University, December 1994.

[Moy89] J. Moy. OSPF. *RFC 1131*, October 1989.

[Moy94a] J. Moy. Multicast Extensions to OSPF. *RFC 1584*, March 1994.

[Moy94b] J. Moy. MOSPF: Analysis and Experience. *RFC 1585*, March 1994.

[Moy94c] J. Moy. Multicast Routing Extensions for OSPF. *Communications of the ACM*, 37(8):61-66, August 1994.

[MS79] P. Merlin and A. Segall. A Failsafe Distributed Routing Protocol. *IEEE Transactions on communications*, 27(9):1280-1287, September 1979.

[MS94] D. Mitzel and S. Shenker. Asymptotic Resource Consumption in Multicast Reservation Styles. *Proc. of ACM SIGCOMM'94*, pp. 226-232, October 1994.

[MTP] Multicast Transport Protocols WWW page, URL http://hill.lut.ac.uk/DS-Archive/MTP.html.

[MV95] S. McCanne and M. Vetterli. Joint Source/Channel Coding for Multicast Packet Video. *Proc. of IEEE International Conference on Image Processing*, October 1995.

[Nag84] J. Nagle. Congestion Control in IP/TCP Internetworks. *RFC 896*. January 1984.

[NEVOT] H. Schulzrinne. *Network Voice Terminal*. Disponible en ftp://gaia.cs.umass.edu/pub/hgschulz/nevot, AT&T/Bell Laboratories.

[NR95] L. Navarro and G. Rodríguez. Aleph: A Large Scale CSCW Environment. *Proc. of JENC'6*, May 1995.

[nv] R. Frederick. *nv, UNIX Manual Pages*. Disponible en ftp://ftp.ee.lbl.gov/nv.tar.Z, Xerox Palo Alto Research Center, USA.

[OB94] J. Ott and C. Bormann. Multicasting the ITU MCS: Integrating Point-to-Point and Multicast Transport, Technical University of Berlin, 1994.

[Pax94] V. Paxton. Growth trends in wide-area TCP connections. *IEEE Network*, 8(4):8-17, July/August 1994.

[PBS89] L.L. Peterson, N.C. Buchholz and R.D. Schlichting. Preserving and Using Context Information Interprocess Communication. *ACM Transactions on Computer Systems*, 7(3):217-246, August 1989.

[Pen93] M.O. Pendergast. Multicast Channels for Collaborative Applications: Design and Performance Evaluation. *ACM SIGCOMM Computer Communications Review*, 23(2):25-38, April 1993.

[PFB95] E. Pastor, D. Fernández and L. Bellido. Cooperative Learning over Broadband Networks. *Proc. of JENC6*, May 1995.

[PGM95] J-J. Pansiot, D. Grad and S. Marc-Zwecker. Towards a Logical Addressing and Routing Sublayer for Internet Multicasting. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, 438-451,October 1995.

[Pos81a] J. B. Postel. Internet Protocol. *RFC 791*, September 1981.

[Pos81b] J. B. Postel. Transmission Control Protocol. *RFC 793*, September 1981.

[Pry91] M. Prycker. *Asynchronous Transfer Mode*. Ellis Horwood Limited, 1991.

[PTK94] S. Pingali, D. Towsley and F. Kurose. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. *Proc. of ACM SIGMETRICS'94*, 14:221-230, 1994.

[Pus93] T. Pusateri. IP Multicast over Token-ring Local Area Networks. *RFC 1469*. June 1993.

[Raj91] B. Rajagopalan. *Failsafe Routing and Multicasting in Dynamic Internets*. PhD Thesis, University of Illinois, Urbana-Champaign, October 1991.

[Raj92] B. Rajagopalan. Reliability and Scaling Issues in Multicast Communication. *Proc. of ACM SIGCOMM'92*, pp. 188-198. August 1992.

[Raj93] B. Rajagopalan. A Mechanism for Scalable Concast Communication. *Computer Communications*, 16(8):485-493, August 1993.

[Ren93] R. V. Renesse. Causal Controversy at Le Mont St-Michel. *Operating Systems Review*, 27(2):44-53, April 1993.

[RJ87] K.K. Ramakrishnan and R. Jain. A Negative Acknowledgement Protocol with Periodic Polling Protocol for Multicast over LANs. *Proc. of IEEE INFOCOMM'87*, pp.502-511, March-April 1987.

[RJ90] K.K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion avoidance in Computer networks. *ACM Transactions on Computer Systems*, 8(2):158-181, May 1990.

[RL93] A. Ravindran and X.T. Lin. Structural Complexity and Execution Efficiency of Distributed Application Protocols. *Proc. of ACM SIGCOMM'93*, 23(4):160-169, September 1993.

[SA83] A. Segall and B. Awerbuch. A Reliable Broadcast Protocol. *IEEE Transactions on Communications*, 31(7):896-901, July 1983.

[Sab95] G. Saba. An Approximate Solution to Minimal-cost Shortest Path, Multipoint Routing Problem. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, pp. 459-466, October 1995.

[SB94] E.T. Saulnier and B.J Bortscheller. Simulation Model Reusability. *IEEE Communications Magazine*, 32(3):64-69, March 1994.

[SBK94] S.B. Shukla, E.B. Boyer and J.E. Klinker. Multicast Tree Construction in Network Topologies with Asymmetric Link Loads. *Technical Report, NPS-EC-04-012, Naval Postgraduate School*, December 1994.

[SFB+87] M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning and L. Suchman. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, 30(1):32-47, January 1987.

[SH95] F. Sennrat and O. Hermanns. Performance Investigations of the MTP - Multicast Transport Protocol. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, pp. 382-392. October 1995.

[Sim94] J.M. Simmons. A Strategy for Designing Error Detection Schemes for General Data Networks. *IEEE Network*, 8(4):41-48, August 1994.

[SL95] Q. Sun and H. Langendorfer. Efficient Multicast Routing for Delay-Sensitive Applications. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, pp. 452-458, October 1995.

[SS95] S. Srinivasan and B. de Supinski. Multicasting in DIS: A Unified Solution. ELECSIM'95, April-June 1995. (También en ftp://ftp.cs.virginia.edu/pub/techreports/CS-95-17.ps.Z)

[TA94] R. Talpade and M.H. Ammar. Single Connection Emulation (SCE): An Architecture for providing a Reliable Multicast Transport Service. *Technical Report GIT-CC-94-47, Georgia Institute of Technology*, October 1994.

[Tan95] A. Tanenbaum. *Distributed Operating Systems*. Prentice-Hall International, 1995.

[TCD95] A.S. Thyagarajan, S.L. Casner and S. Deering. Making the MBONE Real. http://inet.nttam.com, May 1995.

[Toi92] G.F.C. Toinard. A New Way to Design Causally and Totally Ordered Multicast Protocols. *ACM Operating Systems Review*, 26(4):77-83, October 1992.

[Tur94] T. Turletti. The INRIA Videoconferencing System (IVS). *ConneXions*, 3(10), October 1994.

[vat] V. Jacobson and S. McCanne. *vat, UNIX Manual Pages*. Disponible en ftp://ftp.ee.lbl.gov/sun-vat.tar.Z, Lawrence Berkeley Laboratory (LBL), Berkeley, USA.

[VBS95] R. Voigt, R. Barton and S. Shukla. A Tool for Configuring Multicast Data Distribution over Global Networks. http://inet.nttam.com, May 1995.

[Ver93] D.C. Verma. Routing Reserved Bandwidth Multi-point Connections. *Proc. of ACM SIGCOMM'93*, 23(4):96-105, September 1993.

[VRB89] P. Verissimo, L. Rodrigues and M. Baptista. AMp: A Highly Parallel Atomic Multicast Protocol. *Proc. of ACM SIGCOMM'89*, 19(4):83-93, September 1989.

[Wax88] B.M. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617-1622, December 1988.

[wb] V. Jacobson and S. McCanne. *Using the LBL Network whiteboard.* Disponible en ftp://ftp.ee.lbl.gov/wb/sun-wb.tar.Z, Lawrence Berkeley Laboratory (LBL), Berkeley, USA.

[WE94] L. Wei and D. Estrin. The Trade-offs of Multicast Trees and Algorithms. *Proc. of International Conference on Computer Communications and Networks ICCCN'94*, pp. 17-24, September 1994.

[Wei95] L. Wei. *Scalable Multicast Routing: Tree Types and Protocol Dynamics.* PhD Thesis, University of Southern California, December 1995.

[Whe95] B. Whetten. *The Message Bus II and The Reliable Multicast Protocol.* Master's Thesis, University of Illinois at Urbana-Champaign, 1995.

[Win87] P. Winter. Steiner Problem in Networks: A Survey. *Networks*, 17(2):129-167, 1987.

[WMK94] B. Whetten, T. Montgomery and S. Kaplan. A High Performance Totally Ordered Multicast Protocol. *Proc. of the 1994 Dagstuhl Workshop Unifying Theory and Practice in Distributed Systems*, September 1994.

[WPD88] D. Waitzman, C. Partridge and S. Deering. Distance Vector Multicast Routing Protocol. *RFC 1075*. November 1988.

[Xpr95] Xpress Transport Protocol Specification, Version 4, *XTP Forum*, http://www.ca.sandia.gov/xtp/xtp.html. March 1995.

[YGS95] R. Yavatkar, J. Griffioen and M. Sudan. A Reliable Dissemination Protocol for Interactive Collaborative Applications. *Proc. ACM Multimedia'95*, November 1995.

[ZDE+93] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala. RSVP: A new Resource Reservation Protocol. *IEEE Network*, 7(5):8-18, September 1993.

[Zha86] L. Zhang. Why TCP Timers Don't Work Well. *Proc. of SIGCOMM 86*, August 1986.

[ZSM95] T. Ziegler, G. Swieckowski and I. Miloucheva. Mechanismns for Reliable Transmission on High Delay Communication Paths. *Proc. of the second workshop on Protocols for Multimedia Systems PROMS'95*, pp. 170-188, October 1995.